Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Superpixels and Attention for High-quality Object Proposals in Complex Environments

Dissertation

zur Erlangung des akademischen Grades
Dr. rer. nat.
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
der Universität Hamburg

vorgelegt von

Christian Wilms

Hamburg, März 2022

GutachterInnen:
Prof. Dr. Simone Frintrop, Universität Hamburg
Prof. Dr. Jürgen Gall, Universität Bonn

Vorsitzender der Prüfungskommission:
Prof. Dr. Timo Gerkmann, Universität Hamburg

Tag der Disputation: 09. Juni 2022

Abstract

The class-agnostic discovery of objects in images, known as *object proposal generation*, is a fundamental task in computer vision. To discover objects, object proposal generation methods produce boxes or pixel-precise segmentation masks as object proposals. These object proposals support systems for *object detection* or other tasks by focusing the processing on potential objects. Although a trivial task for humans, the class-agnostic discovery of objects in images is challenging for computer vision systems due to the complexity of natural scenes and the large variety of objects. Therefore, object proposal generation is far from being solved. This thesis improves object proposal generation in complex scenes and focuses on two major limitations. First, we address the challenging discovery of small objects that are often missed by existing systems. Second, we propose methods that generate precise objects proposals by improving the adherence of the pixel-precise segmentation masks to the object boundaries. To tackle these problems and highlight the widespread applicability of the proposed solutions, we make major contributions to three areas: object proposal generation, superpixel segmentation, and their applications to real-world tasks.

The first part of the thesis proposes two novel object proposal generation methods based on *Convolutional Neural Networks* (CNNs). First, we address the challenging problem of reliably discovering small objects that are frequently missed by other methods. To alleviate this problem, we propose a new object proposal generation system called *AttentionMask*. AttentionMask uses the concept of visual attention to focus processing on relevant image areas and contains a new module for discovering small objects. This results in a highly efficient approach and improves the results on small objects and objects of all sizes. However, the generated object proposals are coarse and do not adhere well to the object boundaries, similar to other CNN-based methods. To improve the adherence of the object proposals to the object boundaries, we propose *Superpixel-based AttentionMask* (SAM) as an extension of AttentionMask. SAM utilizes highly precise superpixels used in traditional object proposal generation methods to refine AttentionMask's coarse object proposals in an innovative, end-to-end learned framework. This refinement leads to substantially more precise object proposals compared to all existing CNN-based object proposal generation methods and improves the overall results.

In the second part of this thesis, we further improve the adherence of object proposals generated with SAM to the object boundaries. To this end, we propose two superpixel segmentation methods. Both methods generate high-quality superpixel segmentations and reduce the oversegmentation, which negatively impacts the performance of SAM. The first approach extends arbitrary superpixel segmentation methods by utilizing edge detection results to adapt the amount of oversegmentation to the level of detail found in different image areas. The second approach, *DeepFH*, augments the RGB-based superpixel segmentation method by *Felzenszwalb and Huttenlocher* (FH) with CNN-based features learned in a pixel-affinity framework. Since FH limits the oversegmentation by design, the same applies to DeepFH. Both proposed approaches improve the quality of the superpixel segmentations w.r.t. its original versions in multiple directions. For the object proposal generation task, we show

that SAM combined with DeepFH superpixel segmentations outperforms all existing object proposal generation approaches.

The third part of this thesis applies AttentionMask to three challenging, relevant real-world tasks. In the first task, we combine AttentionMask with a tailored classifier to detect airline logos in images. This task is difficult due to a strong influence of adversarial weather effects on the images that challenge the robustness of AttentionMask and the classifier. Second, we utilize AttentionMask in the context of medical instrument segmentation in images acquired during minimally invasive surgeries. This task also poses challenges to the robustness of AttentionMask since effects like smoke or poor illumination severely degrade the images. Finally, we use AttentionMask to localize apples in images depicting complex orchard environments within an agricultural context. The major challenges in this task are the substantial amount of clutter imposed by leaves and the small relative size of the apples. Overall, we show that AttentionMask produces strong detection, segmentation, and localization results on the three tasks. The results also highlight AttentionMask's versatile applicability and its strong robustness.

In summary, we advance the state-of-the-art in object proposal generation by substantially improving the discovery of small objects and by also improving the discovery of objects across all sizes. Moreover, we bridge the gap between CNN-based and traditional superpixel-based object proposal generation methods to produce precise object proposals with strong overall results. These contributions result in high-quality object proposals generated by our systems. To support this process and highlight the general versatility of the systems, we also make contributions to the fields of superpixel segmentation and three application areas.

Zusammenfassung

Die klassenunabhängige *Objektentdeckung* in Bildern (engl. *object proposal generation* oder *object discovery*) ist von grundlegender Bedeutung für verschiedene Bereiche der Bildverarbeitung. So kann ein System zur Objektentdeckung durch die Erzeugung von Objektkandidaten in Form von Boxen oder präzisen Masken den Suchraum für nachfolgende, komplexe Anwendungen wie die *Objekterkennung* (engl. *object detection*) verkleinern und so eine beschleunigte Verarbeitung ermöglichen. Für Menschen ist die klassenunabhängige Entdeckung von Objekten auch in komplexen Szenen mit verschiedensten Objekten mühelos. Bildverarbeitungssysteme stellt diese Aufgabe hingegen vor große Herausforderungen. Daher befasst sich die vorliegende Arbeit mit der Verbesserung der klassenunabhängigen Objektentdeckung. Der Fokus liegt dabei sowohl auf der herausfordernden Entdeckung kleiner Objekte, die von existierenden Systemen zumeist nicht gefunden werden, als auch auf der präzisen Segmentierung der entdeckten Objekte. Dazu werden im Rahmen dieser Arbeit methodische Beiträge in den beiden Bereichen Objektentdeckung und Superpixelsegmentierung vorgestellt. Ferner zeigen weitere Beiträge zu verschiedenen Applikationen die weitreichenden Anwendungsmöglichkeiten der im Rahmen dieser Arbeit präsentierten Methoden.

Der erste Teil der vorliegenden Arbeit stellt zwei neue Systeme zur Objektentdeckung auf Basis von *Convolutional Neural Networks* (CNNs) vor. Zum einen wird die anspruchsvolle Entdeckung kleiner Objekte, die von bestehenden Systemen oft nicht erkannt werden, durch das System *AttentionMask* adressiert. Dazu nutzt AttentionMask das Konzept der visuellen Aufmerksamkeit zur fokussierten Verarbeitung und führt ein neues Modul zur Entdeckung kleiner Objekte ein. Diese Neuerungen führen zu einer effizienten Verarbeitung bei gleichzeitig verbesserten Ergebnissen für die Entdeckung kleiner Objekte sowie Objekte aller Größen. Die so erzeugten Objektkandidaten sind jedoch wie bei anderen CNN-basierten Systemen unpräzise, d.h. die entsprechenden Segmentierungsmasken folgen nur recht grob den eigentlichen Objektkonturen. Daher greift das zweite vorgeschlagene System, *Superpixel-based AttentionMask* (SAM), dieses Problem auf und passt die groben Objektkandidaten von AttentionMask besser an die eigentlichen Objektkonturen an. Dazu verbindet SAM das CNN-basierte AttentionMask mit präzisen Superpixelsegmentierungen, die häufig in klassischen Objektentdeckungssystemen Verwendung finden, zu einem kombinierten, trainierbaren System. Durch diese innovative Kombination ist SAM in der Lage, die Erzeugung präziser Objektkandidaten zu erlernen, die besser an die eigentlichen Objektkonturen angepasst sind. In der Folge übertrifft SAM die Ergebnisse aller vorherigen Objektentdeckungssysteme.

Im zweiten Teil dieser Arbeit werden zwei neue Verfahren zur Superpixelsegmentierung vorgestellt, die noch präzisere Objektkandidaten ermöglichen. Beide Verfahren zielen dabei nicht nur auf eine hohe Qualität der Superpixelsegmentierungen ab, sondern auch auf eine reduzierte Übersegmentierung, was eine positive Wirkung auf die Ergebnisse von SAM hat. Dazu wird im Rahmen dieser Arbeit zum einen ein neues, flexibles Rahmenwerk präsentiert. Es ermöglicht auf Basis von Kantendetektionsergebnissen beliebigen Verfahren zur Superpixelsegmentierung die Generierung von Superpixeln auf relevante, komplexe Bildbereiche zu fokussieren. Dies führt zu einer verminderten Übersegmentierung bei gleichbleibender

oder verbesserter Segmentierungsqualität. Zum anderen wird in dieser Arbeit das neue Superpixelsegmentierungsverfahren DeepFH präsentiert. DeepFH erweitert das RGB-basierte Verfahren von *Felzenszwalb und Huttenlocher* (FH) um gelernte, CNN-basierte Merkmale. Da FH bereits die Übersegmentierung limitiert, ermöglicht DeepFH durch semantisch reichhaltige Merkmale eine Verbesserung der Segmentierungsqualität bei gleichbleibend niedriger Übersegmentierung. In Kombination mit SAM führen beide vorgeschlagenen Systeme zu präziseren Objektkandidaten und verbessern die Ergebnisse von SAM in Bezug auf die ursprünglichen Superpixelsegmentierungen. Die Kombination von SAM und DeepFH übertrifft dabei zudem die Ergebnisse aller anderen existierenden Objektentdeckungssysteme.

Der dritte Teil dieser Arbeit präsentiert den Einsatz von AttentionMask im Kontext von drei komplexen, relevanten Anwendungen. Im Rahmen der ersten Anwendung wird AttentionMask mit einem spezialisierten Klassifikator kombiniert, um Logos von Airlines auf Flugzeugleitwerken zu erkennen. Die Schwierigkeiten dieser Anwendung gehen dabei vor allem mit der durch Wettereinflüsse reduzierten Bildqualität einher. In der zweiten Anwendung wird AttentionMask genutzt, um die Segmentierung von medizinischen Instrumenten in Bilddaten von minimalinvasiven Operationen zu adressieren. Ähnlich wie in der vorherigen Anwendung sind auch hier die Bilder durch äußere Einflüsse wie Rauch, Blut oder suboptimale Beleuchtung beeinträchtigt. Zuletzt wird AttentionMask in einem agrarwirtschaftlichen Kontext zur Lokalisierung von Äpfeln an Apfelbäumen in komplexen Plantagenumgebungen angewendet. Die Lokalisierung wird vornehmlich erschwert durch die große Menge an Blättern, die eine komplexe Bildkomposition zur Folge haben, sowie die geringe relative Größe der Äpfel. Insgesamt ist AttentionMask in der Lage, im Rahmen aller drei Anwendungen vielversprechende und hochwertige Ergebnisse zu erzeugen. Zudem heben die drei Anwendungen die vielseitige Anwendbarkeit sowie die Robustheit von AttentionMask hervor.

Zusammenfassend verbessern die im Rahmen dieser Arbeit vorgestellten methodischen Beiträge und Verfahren die klassenunabhängige Entdeckung von Objekten in einem erheblichen Maße. Dies wird vor allem durch eine verbesserte Entdeckung kleiner Objekte sowie durch präzisere Segmentierungsmasken der Objektkandidaten erreicht. Zudem verknüpfen die Beiträge dieser Arbeit klassische Superpixel-basierte und moderne CNN-basierte Techniken zur Objektentdeckung, was zu einer Aggregation der jeweiligen Stärken und hochqualitativen Objektkandidaten führt. Um diese Verknüpfung zu unterstützen und die vielseitige Anwendbarkeit der Methoden zu demonstrieren, werden im Rahmen dieser Arbeit des Weiteren Beiträge zur Superpixelsegmentierung sowie zu drei Anwendungsgebieten präsentiert.

Publications

Most of the work presented in this thesis has been published in peer-reviewed conference proceedings and established journals of the computer vision community (see below). The code developed for some of the publications is publicly available and can be downloaded from the provided links.

1. Wilms, C.; Frintrop, S.: Edge adaptive seeding for superpixel segmentation. In: German Conference on Pattern Recognition (GCPR), 2017.
   Code available at: `https://www.inf.uni-hamburg.de/en/inst/ab/cv/media/wilms-edge-adaptive-superpixels-v1-0.zip`
2. Wilms, C.; Frintrop, S.: AttentionMask: Attentive, efficient object proposal generation focusing on small objects. In: Asian Conference on Computer Vision (ACCV), 2018.
   Code available at: `https://github.com/chwilms/AttentionMask`
3. Wilms, C.; Heid, R.; Sadeghi, M.A.; Ribbrock, A.; Frintrop, S.: Which airline is this? Airline logo detection in real-world weather conditions. In: International Conference on Pattern Recognition (ICPR), 2020.
4. Wilms, C.; Frintrop, S.: Superpixel-based refinement for object proposal generation. In: International Conference on Pattern Recognition (ICPR), 2020.
   Code available at: `https://github.com/chwilms/superpixelRefinement`
5. Wilms, C.; Frintrop, S.: DeepFH segmentations for superpixel-based object proposal refinement. In: Image and Vision Computing (IMAVIS) 114, 2021

Additionally, two papers were published as preprints on the platform arXiv.

1. Wilms, C.; Johanson, R.; Frintrop, S.: Localizing small apples in complex apple orchard environments. In: arXiv preprint arXiv:2202.11372, 2022
2. Wilms, C.; Gerlach, A.M.; Schmitz, R.; Frintrop, S.: Segmenting medical instruments in minimally invasive surgeries using AttentionMask. In: arXiv preprint arXiv:2203.11358, 2022

Table of Contents

Chapter 1

Introduction

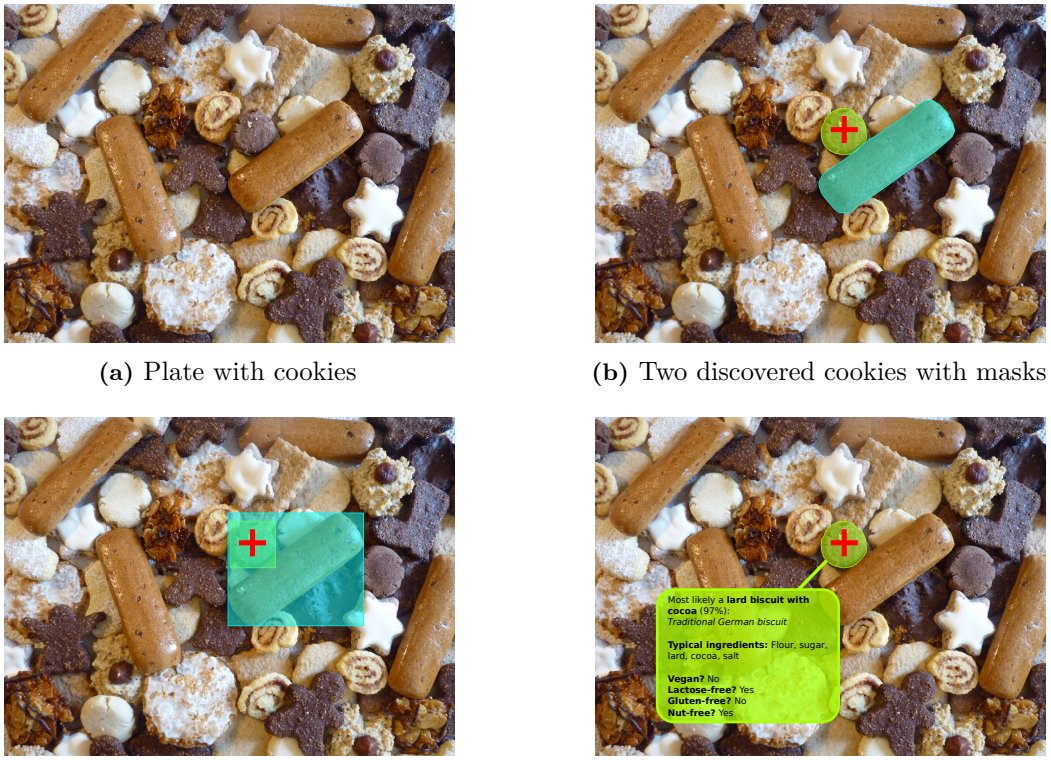
Table of Contents

1.1 Motivation

Imagine it is Christmas time and you are invited to your company's Christmas party. Plates with Christmas cookies like in Fig. 1.1(a) are everywhere, and several cookies are unfamiliar to you. Which cookie fits your vegan diet? How about the small brown one? Let us suppose this scenario happens in 2050, and computationally powerful smart glasses [Lee and Hui, 2018] similar to today's *Microsoft HoloLens 2* are ubiquitous. An app on your smart glasses tracks your gaze and discovers the objects in view as demonstrated in Fig. 1.1(b) for two example objects. This discovery of objects is class-agnostic and does not depend on prior knowledge about the objects. Once the app has discovered the objects, it selects the attended object based on gaze data (red cross in Fig. 1.1). Finally, the app compares the attended object to a database of known objects to gain information about it. In our scenario, a brief description of the cookie, typical ingredients, and other nutrition facts are displayed (see Fig. 1.1(d)). Hence, the brown cookie will not fit your vegan diet.

This scenario sounds fictional, although most components are already available, and similar systems were proposed in simplified contexts [Toyama et al., 2012; Barz and Sonntag, 2016; Barz et al., 2021]. Smart glasses [Lee and Hui, 2018] are a common technique with multiple applications across healthcare [Tepper et al., 2017], robotics [Quintero et al., 2018], or education [Altmeyer et al., 2020]. Several smart glasses such as Microsoft's HoloLens 2 also feature an eye-tracker that estimates the wearer's gaze. Retrieving images of individual objects is a well-studied topic in computer vision [Zheng et al., 2017]. Another fundamental task in this scenario is the precise[1], class-agnostic discovery of objects in images to determine the attended object. The result of this object discovery is a pixel-precise segmentation mask per object since

[1] Precise in this context means that the segmentation mask for a discovered object adheres well to the discovered object's boundary.



(a) Plate with cookies

(b) Two discovered cookies with masks

(c) Two discovered cookies with boxes

(d) Attended cookie with displayed information

Figure 1.1: Example view of a plate with cookies (a) seen through smart glasses and two cookies discovered with masks (b) or boxes (c). The masks allow a better estimation of the attended cookie since they do not overlap substantially. Once the cookies are discovered and the attended cookie is selected based on gaze data (red cross), information about the attended cookie is displayed to the user (d). We will revisit this example in Ch. 10 to showcase the results of this thesis (see Fig. 10.1).

boxes may not allow a distinct localization of the attended object (see red cross in Fig. 1.1(b) and Fig. 1.1(c)). Additionally, the object classes are unknown in such a scenario, leading to a strong demand for generalization. In this thesis, we focus on the described class-agnostic discovery of objects in images, which corresponds to the *object proposal generation* task in computer vision.

For humans, the precise, class-agnostic discovery of objects is effortless, even for 5-months-old infants [von Hofsten and Spelke, 1985]. Similarly, patients with visual form agnosia[2], who are unable to recognize objects reliably, can still perceive them and interact with them [Milner and Heywood, 1989; Goodale et al., 1991]. Hence, humans have a high-level idea of objects, independent of recognizing the object class. Although it is still largely unknown how this process works in humans [Cavanagh, 2011], multiple theories try to explain the perception of objects in human vision. According to the Gestalt theory [Wertheimer, 1922] and the coherence theory [Rensink, 2000], rules of perceptual organization combine basic perceptual units. While Gestalt theory includes local and global cues like symmetry,

[2]Visual form agnosia [Benson and Greenberg, 1969] is a variation of visual agnosia that prevents patients from recognizing the shape of an object. Hence, they are largely unable to reproduce, match, or describe an object's shape. As a consequence, the patients are unable to recognize objects reliably [Benson and Greenberg, 1969].

coherence theory focuses more on local cues like contour continuation [Rensink and Enns, 1995]. Following Rensink [2000], focused attention[3] links such groups of basic perceptual units to create the object perception. However, Rensink [2000] does not provide a relation to the anatomy of the human brain [Cavanagh, 2011].

In computer vision, Sonka et al. [2014] argue that the general task is 'to duplicate the effect of human vision by electronically perceiving and understanding an image'. Since the process of discovering objects is not even fully understood in humans, it is unsurprising that the discovery of objects is far from being solved in computer vision. A significant problem here and in related computer vision tasks is the extraction of high-level semantics from the pixels' intensity or color values. For instance, the type of cookie per pixel is initially unknown to a computer vision system in the introductory example. Those high-level semantics have to be extracted from hand-crafted features [Lowe, 2004; Dalal and Triggs, 2005] or learned from data [Krizhevsky et al., 2012; Ren et al., 2016]. Despite promising results in image classification on clean images [Ioffe and Szegedy, 2015; He et al., 2016a], the results for more complex tasks under previously unseen conditions are below human-level performance [Geirhos et al., 2018b; Liu et al., 2020].

As briefly mentioned above, the computer vision task that corresponds to the class-agnostic discovery of objects is called *object proposal generation*[4]. The goal of object proposal generation methods is to extract every object in the image as a bounding box or a pixel-precise segmentation mask (see Sec. 2.2.1). Unlike object detection or instance segmentation, object proposal generation is not limited to pre-selected object classes or the object classes seen in training. Hence, object proposal generation methods encode general object properties like contrast or a closed boundary [Alexe et al., 2010; Zitnick and Dollár, 2014].

Creating a pixel-precise segmentation mask per discovered object, i.e., segmenting the object from its background, is essential in object proposal generation. Most object proposal generation methods create a pixel-level foreground-background segmentation per discovered object [Carreira and Sminchisescu, 2011; Hu et al., 2017a] or utilize *superpixels* [Ren and Malik, 2003] as an intermediate concept [Uijlings et al., 2013; Manén et al., 2013]. Superpixels are groups of spatially connected pixels sharing a common property like color or texture (see Sec. 2.1) and usually cover object parts. Many approaches for *superpixel segmentation* exist in the literature [Stutz et al., 2018], which are also widely used in computer vision tasks unrelated to object proposal generation [Perazzi et al., 2012; Mostajabi et al., 2015]. Several object proposal generation systems combine multiple superpixels to discover complex objects with pixel-precise segmentation masks [Uijlings et al., 2013; Manén et al., 2013]. This combination leads to a close connection between object proposal generation and superpixel segmentation. Hence, we will cover both topics in this thesis.

Besides the introductory example, which we will revisit in Ch. 10, various computer vision applications utilize object proposal generation methods for the class-agnostic discovery of objects. Originally, object proposals were designed to focus object detection systems on relevant image areas that likely contain objects [Alexe et al., 2010; Girshick, 2015; Ren et al., 2016]. This focused processing prevents object detection systems from processing a multitude of windows[5] extracted from the image that might cover objects [Dalal and Triggs, 2005;

[3] Focused attention is the process of selectively bringing information 'into conscious awareness' [MacKay-Brandt, 2011].

[4] In robotics, object proposal generation is sometimes known as *object discovery* [Frintrop, 2014].

[5] Even small images of size 320×240 lead to more than one billion potential windows [Lampert et al., 2009].

(a) Input image **(b)** Pinheiro et al. [2016] **(c)** Hu et al. [2017a] **(d)** Ground truth

Figure 1.2: Results of the object proposal generation systems by Pinheiro et al. [2016] (b) and Hu et al. [2017a] (c) on an input image (a) with eleven small birds. (d) depicts the ground truth from Lin et al. [2014]. The systems miss most birds due to their small size. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. The circled region is enlarged for visualization. Input image and annotations taken from the COCO dataset [Lin et al., 2014].

Felzenszwalb et al., 2009]. Other computer vision tasks that utilize object proposals include instance segmentation [Hariharan et al., 2014; He et al., 2017a], visual grounding [Rohrbach et al., 2016; Plummer et al., 2018], fine-grained classification [Zhang et al., 2014; He et al., 2018], object tracking [Ošep et al., 2018; Fan and Ling, 2019], or tasks with weak supervision [Bilen and Vedaldi, 2016; Tang et al., 2018]. Similarly, diverse robotics applications employ object proposal generation methods [Chu et al., 2018; Xie et al., 2021]. In such robotics applications, the class-agnostic setting in object proposal generation is of great importance since previously unknown environments and unseen objects are major challenges [Sünderhauf et al., 2018; Xie et al., 2021].

1.2 Limitations of the State-of-the-art

As mentioned above, there is an application-driven demand for methods that generate precise object proposals for arbitrary objects. Suitable superpixel segmentation methods can support this process and improve object proposal generation results. Following the brief introduction of both concepts, we will review some of their major limitations that motivate this thesis.

1.2.1 Object Proposal Generation

Research on object proposal generation has produced several systems either utilizing hand-crafted features [Uijlings et al., 2013; Zitnick and Dollár, 2014; Pont-Tuset et al., 2017] or *Convolutional Neural Networks* (CNNs) [Pinheiro et al., 2015, 2016; Hu et al., 2017a]. Despite the success of those methods, major limitations still remain.

First, discovering small objects[6] is a major challenge in object proposal generation [Zitnick and Dollár, 2014; Pinheiro et al., 2016; Pont-Tuset et al., 2017] as shown in Fig. 1.2. Most small birds are missed (see red contours) by the approaches of Pinheiro et al. [2016] and Hu et al. [2017a], which generate very good results on larger objects though. Since small objects

[6]Small objects cover less than 32^2 pixels in the input image [Lin et al., 2014]. This definition is based on the images of the COCO dataset [Lin et al., 2014] that have an average size of 578×484 pixels (2014 release).

(a) Input image **(b)** Pont-Tuset et al. [2017] **(c)** Hu et al. [2017a]

Figure 1.3: Two examples for the trade-off between precise proposals with low recall (b) and coarse proposals with high recall (c). The proposals in (b) are the result of the superpixel-based approach by Pont-Tuset et al. [2017], while the proposals in (c) were generated with the CNN-based system of Hu et al. [2017a]. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Input images and annotations taken from the COCO dataset [Lin et al., 2014].

are frequently encountered in real-world images due to their original size or the distance to the camera, these misses strongly impede the overall performance. The degraded performance on small objects is mainly related to coarse superpixels or the inherent downsampling process in CNNs. Both effects remove spatial details from the representation of the image, which prevents subsequent parts of the approaches from discovering small objects. Besides the object size, it is largely unknown which object properties negatively influence the object proposal generation results. This lack of knowledge about such object properties is the second limitation of object proposal generation approaches.

Third, existing object proposal generation systems are unable to discover most objects entirely and to segment them precisely. This trade-off between precise proposals with low recall and coarse proposals with high recall is a major challenge in object proposal generation. Traditional[7] systems based on superpixels usually generate precise proposals but frequently miss objects or create proposals covering only object parts (low recall). In contrast, systems utilizing CNNs discover more objects entirely (high recall) but tend to generate coarse proposals due to the inherent downsampling process in CNNs that removes spatial details. Both effects are visible in Fig. 1.3. For instance, Pont-Tuset et al. [2017] use superpixels to generate precise proposals for the people in the foreground of the baseball example while missing the people in the background. In contrast, the CNN-based system by Hu et al. [2017a] discovers all people entirely but only generates coarse segmentation masks that do not adhere well to the peoples' boundaries.

Object proposal generation systems usually generate more proposals than there are objects in the image. The large amount of proposals leads to a high recall and is often preferable for subsequent applications that can not compensate for missed objects [Zitnick and Dollár,

[7]The term traditional denotes computer vision systems that do not utilize deep learning.

2014; Hosang et al., 2015]. To choose only the most promising n proposals, systems generate a ranking of the proposals. However, the quality of the ranking is often questionable [Hosang et al., 2015; Pinheiro et al., 2016; Pont-Tuset et al., 2017] since a generalized object description is difficult to model or learn. Therefore, many proposals are needed to discover all objects compensating for the impaired ranking, which is the fourth limitation of object proposal generation systems.

In summary, the discovery of small objects, the trade-off between high recall and precise proposals, the lack of knowledge about challenging object properties, and the ranking of proposals are the most important limitations in object proposal generation. In this thesis, we focus on the first two limitations while also investigating challenging object properties in object proposal generation. We identified these limitations as the most important ones for the recall-driven [Zitnick and Dollár, 2014; Hosang et al., 2015] object proposal generation task.

1.2.2 Superpixel Segmentation

Similar to the object proposal generation approaches, existing superpixel segmentation methods suffer from several limitations that impact their performance and usability in real-world applications. We will focus our discussion on limitations arising when utilizing superpixels for object proposal generation.

First, many superpixel segmentation methods generate superpixels that are uniformly distributed across the images [Van den Bergh et al., 2012; Achanta et al., 2012; Yao et al., 2015]. Although these approaches produce good results on segmentation benchmarks [Stutz et al., 2018], they ignore the different levels of detail in an image and increase the oversegmentation[8] in semantically uniform areas like objects (see Fig. 1.4). Since recombining several superpixels into one object proposal is challenging, object proposal generation methods prefer superpixel segmentations with less oversegmentation [Uijlings et al., 2013; Manén et al., 2013; Martín García et al., 2015]. However, such superpixel segmentations lack general segmentation quality [Stutz et al., 2018]. This lack of high-quality superpixel segmentations with low oversegmentation is the first limitation of superpixel segmentation methods.

Additionally, most superpixel segmentation methods use simple color or gradient features [Felzenszwalb and Huttenlocher, 2004; Achanta et al., 2012; Van den Bergh et al., 2012]. Few methods employ semantically rich CNN-based features to learn application-specific superpixel segmentations [Tu et al., 2018; Jampani et al., 2018; Yang et al., 2020]. Moreover, these CNN-based methods produce a large amount of oversegmentation. Hence, another limitation is the lack of CNN-based superpixel segmentation methods with low oversegmentation for improved object proposal generation results.

Finally, most superpixel segmentations lack a grid topology. In a grid topology, every superpixel would have a fixed number of neighbors in a fixed spatial layout similar to pixels in an image. The lack of such a fixed layout makes a natural integration of superpixels into CNNs challenging and impedes the combination of both techniques. Few works have tried to circumvent the problem [He et al., 2015; Gadde et al., 2016; Kwak et al., 2017;

[8] Oversegmentation denotes the effect of splitting a region of the desired segmentation into several superpixels. We measure the oversegmentation in terms of Oversegmentation Error (OE) [Stutz et al., 2018] as described in Sec. 2.1.3.

(a) Achanta et al. [2012] **(b)** Yao et al. [2015] **(c)** Felzenszwalb and Huttenlocher [2004]

Figure 1.4: Superpixel segmentations with approximately 100 superpixels using the methods of Achanta et al. [2012] (a), Yao et al. [2015] (b), and Felzenszwalb and Huttenlocher [2004] (c). While many superpixels are wasted in sky or wall regions producing increased oversegmentation in (a) and (b), the result in (c) exhibits less oversegmentation. The reduced oversegmentation leads to a more detailed segmentation of the roof and allows to capture the small windows. Input image taken from the BSD dataset [Martin et al., 2001].

He et al., 2017b; Park et al., 2017], but no general-purpose solution has been presented so far.

Overall, the main limitations of superpixel segmentation methods for object proposal generation are the substantial amount of oversegmentation and the lack of suitable CNN-based approaches within this context. We tackle both limitations in this thesis to improve object proposal generation.

1.3 Scope and Contributions

This thesis advances object proposal generation across all object sizes and addresses several limitations in object proposal generation and superpixel segmentation discussed above. We focus on substantially improving the discovery of small objects and increasing the adherence of the proposals' segmentation masks to the object boundaries. This will lead to high-quality object proposals and is expected to boost subsequent applications like object detection substantially [Hosang et al., 2015]. To support the generation of precise segmentation masks, we also present novel superpixel segmentation methods that limit oversegmentation. Finally, we showcase the versatility and robustness of our systems in three challenging, relevant real-world applications. The most important contributions and the major areas of this thesis are visualized in Fig. 1.5. In the following, we will briefly summarize the major contributions and their relation to the publications associated with this thesis.

Object Proposal Generation

AttentionMask We utilize the concept of human visual attention to develop the new CNN-based object proposal generation system *AttentionMask*. A learned attention focuses the processing on relevant image areas and leads to a more efficient utilization of the limited resources on modern GPUs. Based on the more efficient processing pipeline, we add a dedicated module for discovering small objects that are frequently missed by other methods. The results show that AttentionMask outperforms previous methods in discovering small objects and objects across all sizes while maintaining high computational efficiency. AttentionMask is one of our major methodological contributions and the foundation for our other contributions.

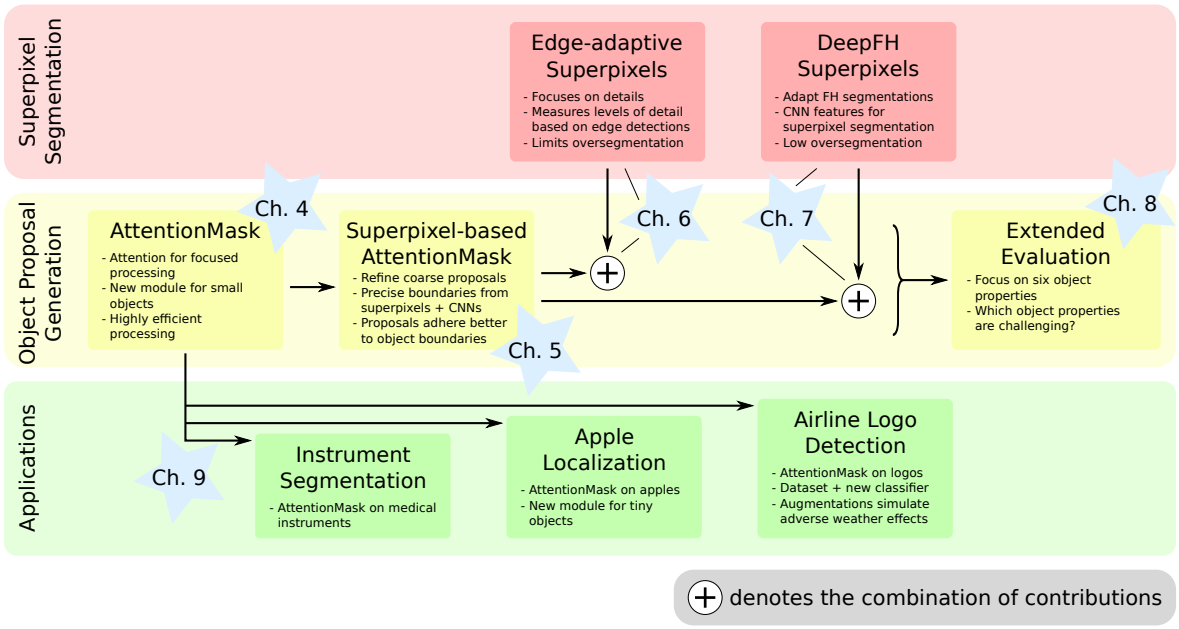


Figure 1.5: Overview of the most important contributions of this thesis assigned to the three major areas superpixel segmentation (red), object proposal generation (yellow), and applications (green). The stars link the contributions to the chapters of this thesis.

We present AttentionMask in Ch. 4, which is based on and extends our publication Wilms and Frintrop [2018].

Superpixel-based AttentionMask Since CNN-based object proposal generation methods lead to coarse object proposals that do not adhere well to the object boundaries, we introduce our innovative *Superpixel-based AttentionMask* (SAM) approach as an extended version of AttentionMask. SAM utilizes high-resolution superpixels and learned features to refine initial coarse AttentionMask proposals. This effectively combines the high recall of CNN-based systems and the ability of superpixel-based approaches to generate precise proposal masks. Our evaluation shows that SAM with FH superpixels [Felzenszwalb and Huttenlocher, 2004] improves the overall object proposal generation results and leads to substantially more precise proposal masks than all previously published CNN-based methods. SAM is another major methodological contribution of this thesis. We present SAM in Ch. 5, which is based on and extends our publications Wilms and Frintrop [2020] as well as Wilms and Frintrop [2021].

Extended Evaluation To assess current object proposal generation systems in more detail and to better understand existing difficulties, we conduct an extended evaluation. This evaluation constitutes the first ever analysis of object proposal generation results w.r.t. the influence of object properties beyond object size. Based on the results, we identify four novel challenges in object proposal generation. We present our extended evaluation in Ch. 8.

Superpixel Segmentation

Edge-adaptive Superpixel Segmentation We present a flexible edge-adaptive superpixel segmentation framework to enhance our object proposal generation system SAM. The framework enables arbitrary superpixel segmentation methods to adapt the distribution of superpixels across the image to the levels of detail in different image areas. To estimate the detail levels, we propose to use the density of edge detection results as a highly effective surrogate measure. Hence, our novel framework allows superpixel segmentation methods to limit oversegmentation that negatively impacts the performance of SAM. We show that utilizing such adapted superpixel segmentations improves the results of SAM compared to non-adaptive superpixel segmentations. We present our edge-adaptive superpixel segmentation framework and the integration with SAM in Ch. 6, which is based on and extends our publication Wilms and Frintrop [2017].

DeepFH Superpixel Segmentation To further improve the superpixel segmentations utilized in our object proposal generation system SAM, we propose the *DeepFH* superpixel segmentation approach. DeepFH is based on the FH superpixel segmentation method [Felzenszwalb and Huttenlocher, 2004], which is utilized by the original formulation of SAM. We extend FH by learning semantically rich CNN-based features in an auxiliary pixel affinity framework. Combining the learned features with the general processing of FH leads to superpixel segmentations that utilize CNN-based features but exhibit less oversegmentation than existing CNN-based approaches. Our evaluation shows an improved superpixel segmentation quality compared to FH. At the same time, the combination of SAM and DeepFH leads to high-quality object proposals outperforming all existing object proposal generation methods. We present DeepFH and its integration with SAM in Ch. 7, which is based on and extends our publication Wilms and Frintrop [2021].

Applications

Airline Logo Detection To showcase the strengths of AttentionMask, we propose an airline logo detection system based on AttentionMask and a specifically tailored classifier to reliably detect airline logos in challenging real-world images. Our evaluation shows that this tailored system is better suited for such a specialized task than application-agnostic object detectors. To increase the robustness w.r.t. the effects of adverse weather conditions in real-world data, we also propose a new learning-free data augmentation scheme that substantially boosts the performance in the absence of suitable training data. We present our airline logo detection system in Sec. 9.1, which is based on our publication Wilms et al. [2020].

Medical Instrument Segmentation We, furthermore, equip AttentionMask with a dedicated post-processing module to address the challenging task of medical instrument segmentation in images acquired during minimally invasive surgeries. The results showcase AttentionMask's flexibility and indicate strong generalization and robustness abilities. Hence, AttentionMask is a very promising foundation for the segmentation of medical instruments during surgeries. We present our medical instrument segmentation system in Sec. 9.2, which is based on our publication Wilms et al. [2022a].

Apple Localization in Orchard Environments As a final application, we utilize AttentionMask to localize apples in complex orchard environments. To locate the small or tiny apples for agricultural applications, we employ a tiling strategy to process an input image in multiple parts. Additionally, we propose a variation of AttentionMask featuring a new module for discovering tiny objects. The results show strong improvements for both approaches over the original AttentionMask and other systems. We present our apple localization system in Sec. 9.3, which is based on our publication Wilms et al. [2022b].

1.4 Outline

The remainder of this thesis is divided into nine chapters (see also Fig. 1.5). In Ch. 2, we present the task definition, datasets, and evaluation measures for superpixel segmentation as well as object proposal generation. Subsequently, we discuss related literature on both topics in more detail in Ch. 3. AttentionMask, our first major methodological contribution, is presented on Ch. 4. This is followed by our second major methodological contribution SAM in Ch. 5. Focusing on superpixel segmentations, Ch. 6 introduces our edge-adaptive superpixel segmentation framework, while Ch. 7 presents our DeepFH superpixel segmentation method. Both chapters also show the combination of the new superpixel segmentations with SAM. After introducing the different object proposal generation systems, our extended evaluation is presented in Ch. 8. The three applications of AttentionMask constitute Ch. 9. Chapter 10 concludes the thesis with a summary of the main findings, a discussion of the presented systems' strengths and weaknesses, and an outlook on potential future research directions.

Chapter 2

Foundations

Table of Contents

To develop methods for object proposal generation and superpixel segmentation as sketched in the introduction, some foundations like the task definition, standard datasets, and evaluation measures are necessary for both tasks. Section 2.1 discusses these foundations for superpixel segmentation, while Sec. 2.2 presents the same topics in the context of object proposal generation. Foundations on CNN backbones utilized in this thesis are discussed in Appendix A.

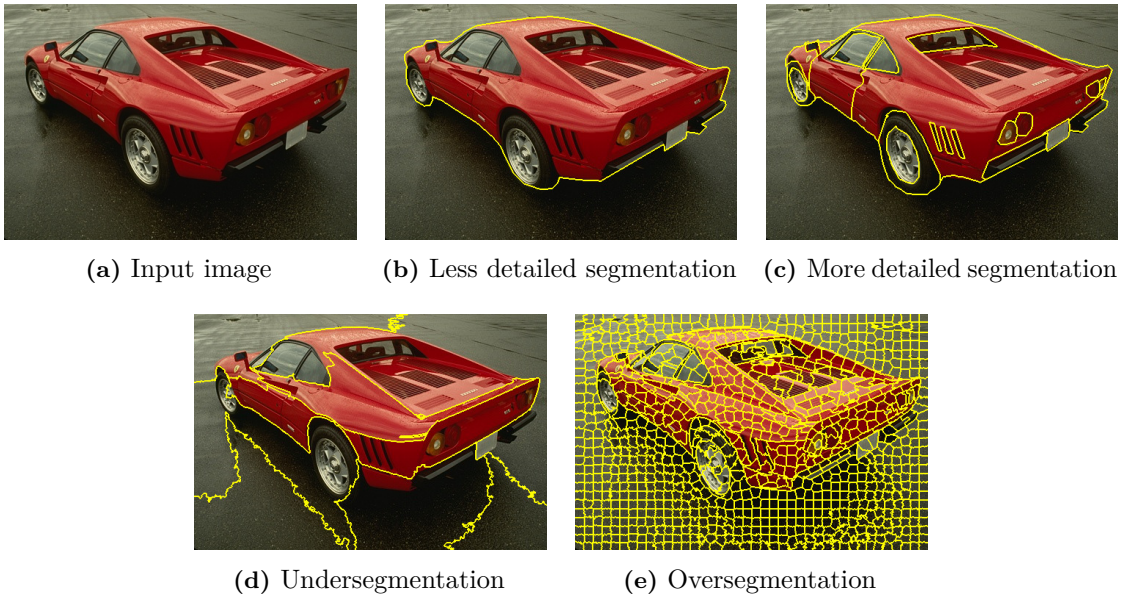
2.1 Foundations on Superpixel Segmentation

Superpixels are one possible building block for object proposal generation systems and led to precise proposals in the past as described in the introduction. Since we present methods for superpixel segmentation and heavily utilize superpixels in this thesis, we introduce terms and general concepts relevant in the context of superpixel segmentation. First, in Sec. 2.1.1, we define superpixels and the superpixel segmentation task. Second, we introduce datasets for superpixel segmentation commonly used in the literature and this thesis in Sec. 2.1.2. To conclude this section, we present common evaluation measures for assessing the quality of superpixel segmentations in Sec. 2.1.3.

2.1.1 Superpixel Segmentation Task

Before defining the superpixel segmentation task, we generally introduce the segmentation of an image. Given an image with height h and width w, we define the set of pixels in the image Ω as

$$\Omega = \{(x, y) : 1 \leq x \leq h \wedge 1 \leq y \leq w\} \subset \mathbb{Z}^2. \tag{2.1}$$



(a) Input image **(b)** Less detailed segmentation **(c)** More detailed segmentation

(d) Undersegmentation **(e)** Oversegmentation

Figure 2.1: Input image (a) with two possible desired segmentations (b) and (c) covering different levels of detail. Both desired segmentations are ground truth annotations from the Berkeley Segmentation Dataset [Martin et al., 2001]. Given the desired segmentation in (b), an example of undersegmentation is visible in (d), while (e) depicts an example of oversegmentation. Input image and annotations taken from the Berkeley Segmentation Dataset [Martin et al., 2001].

An image segmentation is a grouping of the image pixels Ω into regions $S_1, S_2, \ldots$. We denote a segmentation consisting of n regions as $S = \{S_1, \ldots, S_n\}$. However, not every grouping of Ω is a segmentation. Following [Gonzalez and Woods, 2018], the grouping S of the image pixels Ω is a segmentation if S satisfies three conditions:

1. $S_i \cap S_j = \emptyset$ for all pairs of regions in S.
2. $\Omega = \bigcup_{S_i \in S} S_i$.
3. S_i is a connected[1] set of pixels.

Therefore, every pixel is assigned to exactly one region while no region is split into separate parts. Note that Gonzalez and Woods [2018] enforce two additional conditions on a segmentation. However, these conditions are content- or task-specific and unnecessary for a segmentation.

Since this definition only covers the necessary aspects of a segmentation, all four groupings in Fig. 2.1(b) - Fig. 2.1(e) are segmentations. However, not all the segmentations are equally useful, depending on the task. Given the task of segmenting the car from the background, the segmentation in Fig. 2.1(b) would yield a perfect result. In contrast, if the task demands a segmentation of the car's components like doors, windows, and tires, the segmentation in Fig. 2.1(c) is more suitable.

Images are usually arrays of intensity or color values, which makes inferring semantics about certain image regions difficult. For humans, it seems effortless to group the image pixels to form an object or stuff regions as outlined in the introduction. In contrast, computer

[1]The pixels are either 4-connected or 8-connected.

vision systems need multiple complicated processing steps to infer the semantics of the pixels in an image [He et al., 2017a; Chen et al., 2017]. Thus, it is usually impossible to directly infer the final segmentation like in Fig. 2.1(b) or Fig. 2.1(c) using simple color or edge-based heuristics [Felzenszwalb and Huttenlocher, 2004; Levinshtein et al., 2009; Achanta et al., 2012]. Therefore, the generated segmentation will most likely either be too coarse (see Fig. 2.1(d)) or too fine (see Fig. 2.1(e)).

The first effect is known as *undersegmentation*, which results in regions covering multiple objects or stuff areas. Hence the result does not comply with the desired segmentation. Given the simple car segmentation task above, this undersegmentation is visible in Fig. 2.1(d), as multiple regions cover both the car and the background. In contrast, *oversegmentation* is the opposite effect. A region of the desired segmentation is split into several regions. In the context of the simple car segmentation task, Fig. 2.1(e) shows a typical oversegmentation since multiple superpixels cover the car. Note that the definition of over- and undersegmentation depends on the desired segmentation. Additionally, both effects will be measurable (see Sec. 2.1.3) in a typical segmentation. Still, one usually dominates the other.

Representing an image as an oversegmentation leads to an abstraction of the image from tens of thousands or millions of pixels to hundreds or a few thousand regions. For instance, Fig. 2.1(e) shows an example with 933 regions capturing almost all relevant boundaries but splitting objects like the car into many regions. Nevertheless, the segmentation of the car as one region remains possible [Ren and Malik, 2003; Gould et al., 2008]. Ren and Malik [2003] coin the regions of such an oversegmentation *superpixels*, as they represent multiple pixels. The oversegmentation, in this case, is known as the *superpixels segmentation* S with superpixels $S_1, S_2, \dots$ [Liu et al., 2011; Yao et al., 2015].

In addition to the definition above, Ren and Malik [2003] demand the superpixels to be uniform w.r.t. a certain quality. This property is automatically satisfied given a reasonable superpixel segmentation method. Some authors add more constraints to the definition of superpixels. For instance, Stutz et al. [2018] demand a controllable number of superpixels, while Ren and Malik [2003] and Achanta et al. [2012] demand a spatially uniform distribution of superpixels. As discussed in the introduction, these additional constraints might not be useful for subsequent applications. For instance, to properly segment the car in Fig. 2.1, a detailed oversegmentation of the uniform background is not necessary.

Since superpixels are the result of an oversegmentation, their shape is not given by the sensor's technical properties, like in the case of pixels. A superpixel's shape is purely driven by the image content and the superpixel segmentation method. Hence, superpixels can represent arbitrarily shaped image areas. Combined with the reduced number of basic entities, it enables faster processing based on superpixels while retaining accurate results. Therefore, several computer vision systems utilize superpixels across different applications [Perazzi et al., 2012; Uijlings et al., 2013; Mostajabi et al., 2015].

Overall, an oversegmentation assigns the image pixels to regions while maintaining the relevant, task-specific structures of the image. The regions of this oversegmentation or superpixel segmentation are called superpixels and allow an abstract but precise representation of the image content with less basic entities compared to the image pixels.

2.1.2 Superpixel Segmentation Datasets

To train and evaluate superpixel segmentation methods, common datasets are inevitable. Such datasets define regions in an image that should be distinguished or boundaries that a superpixel segmentation should contain. In general, datasets used for evaluating and training superpixel segmentation methods consist of images with at least one manually created segmentation. Some datasets are explicitly designed for evaluating superpixel segmentation methods [Martin et al., 2001; Gould et al., 2009], while others were originally designed for a specific task [Yamaguchi et al., 2012; Cordts et al., 2016]. This difference in purpose leads to different styles of annotations. In the dataset proposed by Martin et al. [2001], the annotators are only instructed to segment the image into regions, which leads to a diverse set of annotations. In contrast, Yamaguchi et al. [2012] propose a dataset for clothing parsing[2], which leads to a clear definition of object classes that should be segmented. Hence, the background or objects of classes that the authors did not define were not segmented.

The most common dataset used in superpixel segmentation is the *Berkeley Segmentation Dataset* (BSD) [Martin et al., 2001]. Most other datasets are used by a few authors or in application-specific contexts. In the following, we describe the five datasets utilized in the benchmark of Stutz et al. [2018] in more detail since they are used in this thesis. Stutz et al. [2018] argue that this selection of datasets incorporates different application areas of superpixel segmentation methods like indoor scenes, outdoor scenes, and scenes with people.

Berkeley Segmentation Dataset

The *Berkeley Segmentation Dataset* (BSD) [Martin et al., 2001] contains 200 training images, 100 validation images, and 200 test images of size 481×321 (version BSD-500). The images depict natural scenes with at least one object, leading to a large variety of scenes. Human annotators were asked to segment the images into regions representing things without more detailed instructions for generating annotations. Per image, annotations created by five different annotators are available to reflect different granularities. Figure 2.2 shows two example images from the BSD test set with two annotations each. The difference in granularity between the two depicted annotations per example is visible in the faces (upper example) and the houses (lower example).

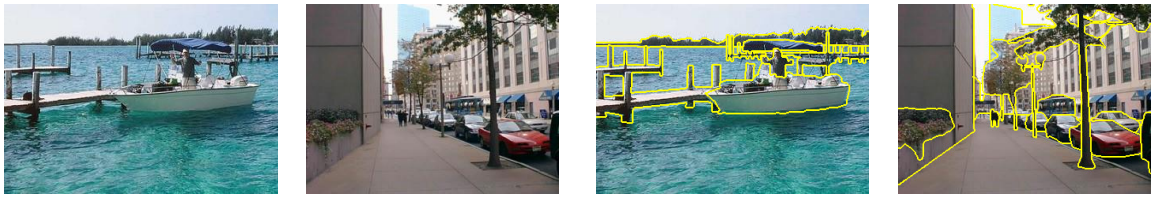
Stanford Background Dataset

Gould et al. [2009] proposed the *Stanford Background Dataset* (SBD) with 238 training images and 477 test images (split proposed by Stutz et al. [2018]) of around 320×240 pixels. The images were collected from different datasets [Criminisi, 2004; Hoiem et al., 2007; Everingham et al., 2010; Russell et al., 2008] and contain outdoor scenes with at least one prominent object. In comparison to the BSD dataset, the images are more complex. Gould et al. [2009] created the dataset to decompose an image into semantically meaningful parts and reason about their geometry. Therefore, the annotations focus on entire objects and stuff regions. In

[2]Clothing parsing is the task of classifying and segmenting the garments that a person wears [Yamaguchi et al., 2012].

(a) Dataset images **(b)** Less detailed annotations **(c)** More detailed annotations

Figure 2.2: Two example images (a) of the BSD dataset [Martin et al., 2001] with annotations of different granularity. The annotations in (b) capture entire objects, while the annotations in (c) also segment individual object parts. Base images and annotations taken from the BSD dataset [Martin et al., 2001].



(a) Dataset images **(b)** Annotations

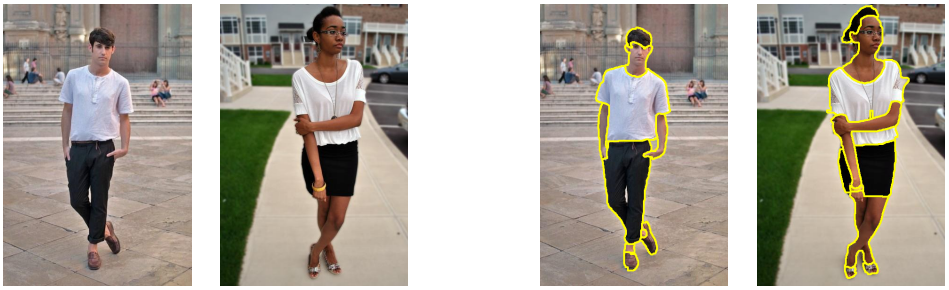
Figure 2.3: Two example images (a) with annotations (b) from the SBD dataset [Gould et al., 2009]. The annotations are detailed but do not segment object parts. Base images and annotations taken from the SBD dataset [Gould et al., 2009].

contrast to the BSD dataset, only one annotation per image is available. Figure 2.3 shows two examples of the SBD test dataset with annotations. It is visible from both examples that the annotations are detailed, although they do not capture object parts as in the BSD dataset.

Fashionista Dataset

Yamaguchi et al. [2012] collected the *Fashionista Dataset* (Fash) with 222 training images and 463 test images (split proposed by Stutz et al. [2018]). The 400×600 images were published by users on a fashion website[3] and show people presenting their outfits in front of various backgrounds. Since Yamaguchi et al. [2012] focus on clothing parsing, the task during annotation was to segment 53 different types of garments or accessories as well as hair and skin. For skin areas, different body parts like hands and legs are not separated when touching (see right example in Fig. 2.4(b)). The background is not segmented further, even if it contains objects. Thus, the dataset is relatively simple compared to the datasets

[3]CHICTOPIA: `https://www.chictopia.com`



(a) Dataset images **(b)** Annotations

Figure 2.4: Two example images (a) and annotations (b) from Fash dataset [Yamaguchi et al., 2012]. The background is not annotated, since the annotations focus on the person, the garments, and the accessories. Base images and annotations taken from the Fash dataset [Yamaguchi et al., 2012].

(a) Dataset images **(b)** Annotations

Figure 2.5: Two example images (a) with annotations (b) from the NYU dataset [Silberman et al., 2012]. The images show typical scene compositions of the dataset, covering cluttered indoor environments with poor lighting conditions (left scene) and missing annotations of background objects (right scene). Base images and annotations taken from the NYU dataset [Silberman et al., 2012].

BSD and SBD. Like the SBD dataset, the Fash dataset provides only one annotation per image. Figure 2.4 visualizes two example images with annotations from the Fash dataset, showing the focus of the annotations on the peoples' garments and ignoring background regions.

NYU Depth Dataset V2

Silberman et al. [2012] proposed the *NYU Depth Dataset V2* (NYU) that comprises RGB-D images of indoor scenes. Silberman et al. [2012] aim to provide a dataset for studying relations of objects in scenes from lived-in rooms that might be cluttered or provide suboptimal lighting conditions. Overall, the dataset contains 1449 images of size 640×480 covering a variety of apartments and public indoor places. We use 399 images for testing and 199 images for training in this thesis as proposed by Stutz et al. [2018]. Similar to the Fash dataset, the annotation process focused on segmenting objects. Figure 2.5 shows two example images with annotations from the NYU test set. The poor lighting conditions are well visible in the left example of Fig. 2.5, while the right example shows a cluttered scene with many unannotated objects on the shelf in the background.

SUN RGB-D Dataset

Similar to the NYU dataset, Song et al. [2015] provide the *SUN RGB-D Dataset* (SUN) with 10.335 RGB-D images. The images were collected from various datasets [Silberman et al., 2012;

(a) Dataset images **(b)** Annotations

Figure 2.6: Two example images (a) and annotations (b) from the SUN dataset [Song et al., 2015] showing typical cluttered indoor scenes. The annotations highlight typical problems, including imprecise or missing annotations and merged instances. Base images and annotations taken from the SUN dataset [Song et al., 2015].

Xiao et al., 2013; Janoch et al., 2013] and have sizes between 561×427 and 730×530 pixels. Following the split of Stutz et al. [2018], we use 400 images for testing and 200 images for training in this thesis, which are not included in the NYU dataset. The challenges w.r.t. clutter and lighting conditions are similar to the NYU dataset. The SUN dataset provides annotations for objects of roughly 800 classes. However, the segmentation quality is worse compared to the other datasets, since several objects are only coarsely segmented. Figure 2.6 shows two example images of the dataset with annotations. From the left example in Fig. 2.6, it is visible that some objects like the pillows or the sofa are only coarsely captured. Furthermore, instances of the same class are not always distinguished when overlapping on the image plane, like the pillows in the left scene. Similar to the NYU dataset, some objects are not annotated, like the statues or the boxes in the right scene in Fig. 2.6.

2.1.3 Superpixel Segmentation Evaluation

Evaluation measures are necessary to assess the quality of superpixel segmentation methods w.r.t. the annotated ground truth. In general, superpixels should adhere to all annotated boundaries in an image and, in an ideal case, not add additional, artificial boundaries. Over time, various measures for evaluating a superpixel segmentation S given a ground truth annotation G have been proposed. Note that G is a segmentation of the image similar to S but was generated manually. The most commonly used measure is *Boundary Recall* (BR) [Martin et al., 2004] that assesses how many boundary pixels[4] of G are covered with the boundary pixels of S given a certain tolerance. We discuss BR in more detail below. An extension of this is the *Mean Distance to Edge* (MDE) [Benesova and Kottman, 2014] that measures the distance between each boundary pixel in G and the closest boundary pixel in S.

To determine how many artificial boundaries a superpixel segmentation introduces, Martin et al. [2004] propose *Boundary Precision* (BP). In this context, artificial boundaries denote boundaries that are not included in the ground truth annotation. The *Oversegmentation Error* (OE) [Stutz et al., 2018] measures a similar effect by calculating the amount of oversegmentation in S as detailed below. To determine the opposite quality of a superpixel segmentation, Levinshtein et al. [2009] first proposed the *Undersegmentation Error* (UE). UE measures the amount of undersegmentation by assessing the leakage of superpixels across the boundaries of G (see below). Also assessing the undersegmentation, *Achievable Segmentation*

[4]Boundary pixels are pixels of a region in a segmentation that are 4-connected or 8-connected to at least one pixel on another region of the segmentation.

Accuracy (ASA) [Liu et al., 2011] models G with the superpixels in S and measures the leakage of the superpixels across the boundaries in G.

Throughout this thesis, we use Boundary Recall (BR), Undersegmentation Error (UE), and Oversegmentation Error (OE). These measures are commonly used (BR and UE) and cover all qualities of superpixel segmentations, while most other measures are redundant (BR is similar to MDE, OE is similar to BP, and UE is similar ASA). We also use the *Overall Segmentation Quality* (OSQ) as a combination of BR and UE following Stutz et al. [2018]. For a fair comparison of superpixel segmentation methods, we compare the superpixel segmentations based on the average number of superpixels across the dataset. Otherwise, a practically irrelevant superpixel segmentation with one pixel per superpixel would also recall all annotated boundaries. If multiple annotations are available, we follow Stutz et al. [2018] and pick the worst value per measure and image for averaging. In the following we discuss the measures used in this thesis in more detail.

Boundary Recall

Boundary Recall (BR) [Martin et al., 2004] measures how many boundary pixels of the ground truth annotation G are covered by the boundary pixels of the superpixel segmentation S given a tolerance ϵ. To compute BR, we define True Positives ($\text{TP}(S, G, \epsilon)$) as the boundary pixels of G, whose L_1 distance to any boundary pixel in S is smaller than ϵ. Similarly, False Negatives ($\text{FN}(S, G, \epsilon)$) are boundary pixels of G, whose L_1 distance to any boundary pixel in S is greater or equal ϵ. TP and FP are visualized in Fig. 2.7(c), where the blue line is the annotation boundary. The parts of the boundary covered by a superpixel boundary within a distance of ϵ are marked green (TP), while those not covered are marked red (FN). Thus, given a superpixel segmentation S, a ground truth annotation G, and a tolerance ϵ, BR is defined as

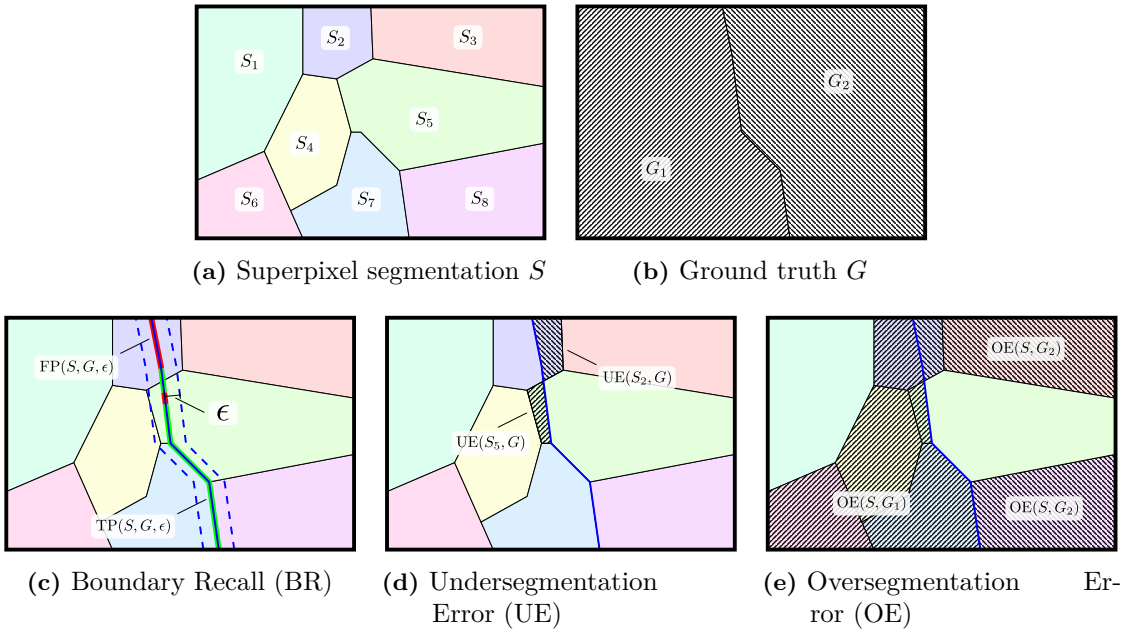
$$\text{BR}(S, G, \epsilon) = \frac{\text{TP}(S, G, \epsilon)}{\text{TP}(S, G, \epsilon) + \text{FN}(S, G, \epsilon)}. \tag{2.2}$$

A larger BR denotes better results since more annotated boundaries are captured. However, a superpixel segmentation with every superpixel representing exactly one pixel would score a perfect BR, although being useless for subsequent processing. Following Stutz et al. [2018], we set $\epsilon = \lfloor 0.5 + 0.0025 \cdot \sqrt{h^2 + w^2} \rfloor$ for an image of height h and width w to adapt the BR to the image size.

Undersegmentation Error

The *Undersegmentation Error* (UE), first introduced by Levinshtein et al. [2009], measures the leakage of the superpixels in S across the boundaries of the ground truth annotation G. Hence, UE assesses the error introduced by undersegmentation in S w.r.t. G. Figure 2.7(d) visualizes an example for the UE given the superpixel segmentation S in Fig. 2.7(a) and the ground truth annotation G in Fig. 2.7(b). Superpixel S_2 (bluish, top center in Fig. 2.7(d)) leaks across the blue contour between the regions G_1 and G_2 of the ground truth annotation. Similarly, superpixel S_5 (green, center in Fig. 2.7(d)) leaks across the blue contour and into G_1. Both cases are undersegmentation errors.

To measure the errors, different definitions [Achanta et al., 2012; Neubert and Protzel, 2012; Van den Bergh et al., 2012] exist since the original formulation by Levinshtein et al. [2009] creates UEs that are not bound to the range $[0, 1]$. Additionally, Achanta et al. [2012] and



(a) Superpixel segmentation S **(b)** Ground truth G

(c) Boundary Recall (BR) **(d)** Undersegmentation Error (UE) **(e)** Oversegmentation Error (OE)

Figure 2.7: Superpixel segmentation S consisting of eight superpixels $S_1, \ldots, S_8$ (a) and ground truth annotation G with regions G_1 and G_2 (b). (c) shows the Boundary Recall (BR) for S given G as the part of the ground truth boundary between G_1 and G_2 (solid blue contour) that is covered by superpixel boundaries in S within a tolerance of ϵ. These parts of the boundary are highlighted in green. (d) presents the Undersegmentation Error (UE) as the leakage of S_2 into G_2 and S_5 into G_1 (hatched areas). Similarly, (e) denotes the Oversegmentation Error (OE) as the areas of G_1 and G_2 that are not covered by the superpixel with the largest overlap (hatched areas).

others pointed out that the definition of Levinshtein et al. [2009] strongly penalizes superpixels leaking only slightly across the annotated boundaries. Circumventing both problems, Neubert and Protzel [2012] propose a UE that accumulates the smaller part of the intersection and the difference of S_j and G_i. This corresponds to the hatched areas in Fig. 2.7(d). We follow this definition of Neubert and Protzel [2012] in this thesis. Hence, UE is defined as:

$$\mathrm{UE}(S, G) = \frac{1}{h \cdot w} \sum_{G_i \in G} \sum_{S_j \in S} \min\{\|S_j \cap G_i\|, \|S_j \setminus G_i\|\}. \tag{2.3}$$

Since UE describes an error, smaller values for UE denote better superpixel segmentation results. Similar to BR, a superpixel segmentation with one superpixel per pixel would score a perfect UE but lead to severe oversegmentation, making it practically useless.

Overall Segmentation Quality

Since BR and UE measure different properties of a superpixel segmentation, a combined measure is useful to assess the segmentation quality with a single number. For this purpose, Stutz et al. [2018] propose the *Overall Segmentation Quality* (OSQ) as a weighted sum of BR and the inverse of UE:

$$\mathrm{OSQ}(S, G, \epsilon) = \gamma\, \mathrm{BR}(S, G, \epsilon) + (1 - \gamma)(1 - \mathrm{UE}(S, G)). \tag{2.4}$$

We follow Stutz et al. [2018] and set $\gamma = 0.5$ while following the same calculation for ϵ as described for BR. Overall, OSQ reflects both the recall of boundary pixels and the errors induced by undersegmentation in a unified measure. We will use OSQ throughout the thesis to optimize the parameters of superpixel segmentation methods.

Oversegmentation Error

In contrast to BR and UE, the *Oversegmentation Error* (OE) [Stutz et al., 2018] assesses the errors introduced by oversegmentation. To measure this error, for each region $G_i \in G$, the superpixel $S_j \in S$ with the largest overlap is identified. In the example in Fig. 2.7(e), G_1 has the largest overlap with S_1, while G_2 has the largest overlap with S_5. The remainder of G_1 or G_2, i.e., the part of G_1 or G_2 not covered by S_1 or S_5, are the oversegmentation errors (hatched areas in Fig. 2.7(e). Thus, for an image with $h \cdot w$ pixels, the OE is defined as

$$\mathrm{OE}(S, G) = \frac{1}{h \cdot w} \sum_{G_i \in G} \min_{S_j \in S} \{\|G_i\| - \|S_j \cap G_i\|\}. \tag{2.5}$$

Since OE denotes an error, small values correspond to better results. OE is rarely used for evaluating superpixel segmentations, since the goal of a superpixel segmentation is to oversegment the image. However, for certain applications it is important to compare the strength of oversegmentation (see Ch. 5 - Ch. 7).

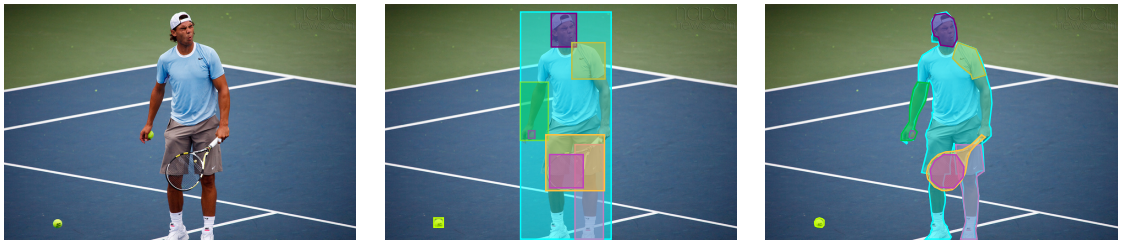
2.2 Foundations on Object Proposal Generation

The second major task we address in this thesis is object proposal generation. In the introduction, we briefly described the object proposal generation task as the discovery of arbitrary objects in an image. This section provides details on the task (see Sec. 2.2.1), relevant datasets (see Sec. 2.2.2), and dedicated evaluation measures (see Sec. 2.2.3).

2.2.1 Object Proposal Generation Task

The object proposal generation task was first[5] introduced by Alexe et al. [2010]. They described the task in the context of object detection. Instead of processing all possible windows of an image, object detection systems, which localize and classify objects, should only process image patches that fully contain one object. This selection of relevant locations (windows or masks), called *object proposals*, is known as the *object proposal generation* task. Figure 2.8 shows an example for an object proposal generation result on an image with large background areas. The object proposals (boxes in Fig. 2.8(b), masks in Fig. 2.8(c)) cover only objects or object parts and reduce the search space for subsequent methods while also removing false positives [Alexe et al., 2012]. Hence, a suitable object proposal generation method increases the efficiency of subsequent object detectors [Alexe et al., 2012] and improves the results [Girshick, 2015]. Since the original formulation by Alexe et al. [2010], other applications have utilized object

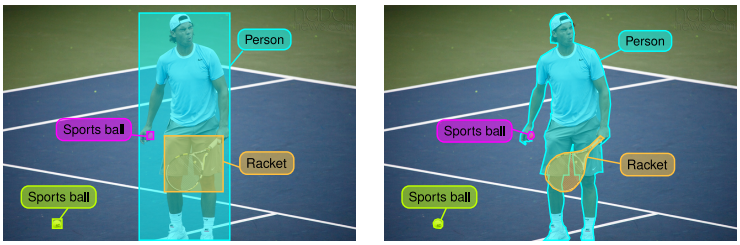
[5] Earlier approaches [Russell et al., 2006; Malisiewicz and Efros, 2007; Gu et al., 2009] followed a similar basic idea, reducing the number of possible windows for object detection. However, they proposed simply taking individual superpixels or all possible small groups of superpixels as proposals without further classification, feature-based merging, or ranking of proposals.



(a) Input image

(b) Box-based object proposal generation result

(c) Mask-based object proposal generation result



(d) Object detection result

(e) Instance segmentation result

Figure 2.8: Input image (a) with manually generated results for box-based (b) or mask-based (c) object proposal generation. The results discover the four major objects (person, racket, and two sports balls) and some object parts (head, leg, arm, racket strings) without further classification. In contrast, the manually generated object detection (d) and instance segmentation (e) results include classifications of the objects and are fixed to a set of pre-selected or learned classes. Input image taken from the COCO dataset [Lin et al., 2014].

proposals such as instance segmentation [Hariharan et al., 2014; He et al., 2017a], visual grounding [Rohrbach et al., 2016; Plummer et al., 2018], fine-grained classification [Zhang et al., 2014; He et al., 2018], object tracking [Ošep et al., 2018; Fan and Ling, 2019], or tasks with weak supervision [Bilen and Vedaldi, 2016; Tang et al., 2018].

Formally, object proposal generation is the task of

1. composing a region (object proposal) for every object in the image
2. with as few regions (object proposals) as possible and
3. additionally, producing a ranking of the regions (object proposals).

The regions or object proposals are either bounding boxes (see Fig. 2.8(b)) or pixel-precise segmentation masks (see Fig. 2.8(c)). We denote the different formulations as box-based and mask-based object proposal generation. The mask-based object proposals precisely segment the object and allow an analysis of the object's shape in subsequent applications [Vicente et al., 2011]. In contrast, box-based proposals only allow a rough estimation of the object's location and extent. As mentioned in the last part of the task definition, a ranking of the object proposals is required. This ranking allows for prioritized processing in case of limited computational resources and is used for evaluation purposes. To generate a ranking, object proposal generation systems produce an *objectness score* per object proposal. The objectness score is usually in the interval of $[0, 1]$ and reflects the chance of an object proposal to fully contain one object [Alexe et al., 2010]. Alternatively, some methods limit the number of object proposals using parameters [Krähenbühl and Koltun, 2014; Rantalankila et al., 2014].

Object proposals aim to discover all objects in an image as described above. This is also independent of the object classes represented in the training set and makes object proposal generation a class-agnostic task. Thus, a general definition of the term *object* is necessary. Additionally, it is also necessary to differentiate objects from other structured background elements like sky or street surface, often coined *stuff* [Forsyth et al., 1996]. For defining objects, different definitions exist [von Hofsten and Spelke, 1985; Forsyth et al., 1996; Alexe et al., 2010]. We follow the definition of von Hofsten and Spelke [1985] from psychology and refer to objects as 'manipulable units with internal coherence and external boundaries'.

The class-agnostic setting of the object proposal generation task with the high-level definition of objects also distinguishes it from related computer vision tasks like object detection and instance segmentation. Object detection [Ren et al., 2016; Redmon et al., 2016] describes the task of localizing and classifying all objects in an image but is fixed to pre-selected object classes or the object classes seen in training. The localization in object detection operates on the level of boxes as visible from the example in Fig. 2.8(d). Instance segmentation [Hariharan et al., 2014; He et al., 2017a] is a similar task and differs from object detection only in the localization based on pixel-precise masks (see Fig. 2.8(e)). Hence, object detection and instance segmentation are similar to object proposal generation. However, those tasks are not class-agnostic, since they are fixed to a set of object classes.

2.2.2 Object Proposal Generation Datasets

Similar to the datasets for superpixel segmentation, we present commonly used datasets for training and evaluating object proposal generation methods. In general, datasets for object proposal generation need annotations for all objects or objects of a pre-defined set of object classes per image. Note that object proposal generation systems do not utilize the class information. As mentioned above, the annotations are either boxes or pixel-precise masks. Due to the substantial manual effort to create mask annotations, most datasets focus on boxes.

The most commonly used datasets in object proposal generation [Alexe et al., 2010; Gidaris and Komodakis, 2016; Pinheiro et al., 2016] are the datasets from the *PASCAL Visual Object Classes* (PASCAL VOC) challenges [Everingham et al., 2010] with box and some mask annotations as well as the *Microsoft Common Objects in Context* (COCO) dataset [Lin et al., 2014] with box and mask annotations. Since this thesis focuses on object proposal generation with pixel-precise masks utilizing CNNs, the PASCAL VOC datasets are not large enough. The largest PASCAL VOC dataset with mask-based annotations covers only 6929 objects (training set and validation set of the 2012 release), while the COCO dataset comprises 896,782 objects with mask-based annotations (training set and validation set of the 2014 release). Hence, we utilize the COCO dataset and the recently introduced *Large Vocabulary Instance Segmentation* (LVIS) dataset [Gupta et al., 2019] in this thesis. The following sections describe these two datasets in more detail. Besides the mentioned datasets, few authors use other datasets like variations of the ImageNet dataset [Russakovsky et al., 2015] or the previously introduced datasets NYU and SUN, among others.

Microsoft Common Object in Context Dataset

Lin et al. [2014] proposed the *Microsoft Common Objects in Context* (COCO) dataset for various recognition tasks, including object detection, keypoint detection, or panoptic

segmentation. The dataset contains 328,000 images with mask and box annotations for objects of 80 classes. The classes cover indoor and outdoor objects, including animals, vehicles, or food. The complexity of the images in the COCO dataset is high, with few iconic object views and many annotated small objects. On average, each image contains 7.27 annotated objects, while an annotated object covers 4.33% of the image (training set and validation set of the 2014 release).

The 2014 release used throughout this thesis has 82,783 images for training. Additionally, 40,504 images for validation were released, including annotations. Since the annotations for the original test data are not publicly available, it is best practice to use the first 5000 validation images for testing [Pinheiro et al., 2016; Hu et al., 2017a]. Thus, we denote the first 5000 images of the original validation set as the test set. From the remaining original validation images, we randomly select 5000 images as our validation set following [Hu et al., 2017a]. Across the dataset, the images have varying sizes and aspect ratios, with an average height of 484 pixels and an average width of 578 pixels (2014 release).

The object masks in the COCO dataset were manually annotated by first identifying an instance and later creating the mask annotations. The masks consist of a coarse polygon, which leads to imprecise annotations that do not adhere to all details of the object boundaries. Figure 2.9 shows two example images with annotations from the COCO dataset. It is visible from the annotations in the left example that the coarse polygons lead to a sometimes poor quality of the annotated masks. For instance, the mask annotating the zebra in the back does not cover the zebra's head. The right example highlights the high complexity of the dataset with multiple different annotated and unannotated objects.

Since the COCO dataset is complex and sufficiently large for training and evaluating deep learning-based systems, we use the 2014 release of the COCO dataset in this thesis.

Large Vocabulary Instance Segmentation Dataset

To foster instance segmentation research, Gupta et al. [2019] present the complex *Large Vocabulary Instance Segmentation* (LVIS) dataset. The dataset consists of the images from the COCO dataset and adds more precise and diverse annotations based on a multi-stage annotation and verification pipeline. The more precise annotations allow a better assessment of precise object segmentations in tasks like instance segmentation or object proposal generation [Gupta et al., 2019]. The precise annotations are highlighted by the left example in Fig. 2.10(b), which depicts the same example as Fig. 2.9(b). Compared to the COCO annotations, the zebras are precisely captured, including both heads and even details like the ears.

Moreover, Gupta et al. [2019] do not limit the annotations to 80 object classes but use a common vocabulary of 1723 object classes to annotate the LVIS dataset. This large set of object classes makes the LVIS dataset more general and challenging than the COCO dataset. Since it is difficult to annotate objects of 1723 classes consistently throughout thousands of images, coherence is only enforced within each image. Moreover, frequently occurring classes are removed from several images to balance the distribution of classes to some extent. As a result, frequently occurring classes like *person* are not annotated in every image. The more detailed annotations are visible in the right scene in Fig. 2.10(b). Compared to the annotations in the COCO dataset (see Fig. 2.9(b)), the LVIS annotations cover significantly

(a) Dataset images **(b)** Annotations

Figure 2.9: Example images from the COCO dataset [Lin et al., 2014] (a) with mask annotations (b). The left example shows typical coarse polygon annotations that do not adhere well to the object boundaries. The right example shows a typical complex scene. Nevertheless, only objects from the pre-defined classes are annotated. Base image and annotations taken from the COCO dataset [Lin et al., 2014].

(a) Dataset images **(b)** Annotations

Figure 2.10: Images from the LVIS dataset [Gupta et al., 2019] (a), which are taken from the COCO dataset, with more precise and complete annotations (b). Compared to the COCO dataset (see Fig. 2.9(b)), the annotations in the LVIS dataset capture the object boundaries more precisely (left scene) and cover more objects (right scene). Base image and annotations taken from the LVIS dataset [Gupta et al., 2019].

more objects. For instance, individual pillows, lamps, and paintings, which are not part of the 80 classes in the COCO dataset, are precisely annotated.

Since the images are identical to the COCO dataset, the image statistics are also identical. However, due to the new annotations, 12.29 objects are annotated on average per image. Moreover, the average relative size per annotation drops to 1.74%. Hence, each image contains more objects that are much smaller compared to the COCO dataset. The LVIS dataset also uses a different split of images leading to 100,170 training images, 19,809 validation images, and 19,822 test images without publicly available annotations. We use the first 5000 images of the LVIS validation set in this thesis to evaluate precise object proposals in Ch. 5 - Ch. 7 and denote it as the LVIS test set. Another 5000 images of the original LVIS validation set comprise our LVIS validation set, similar to the COCO dataset.

2.2.3 Object Proposal Generation Evaluation

In Sec. 2.1.3, we discussed evaluation measures that determine how well a superpixel segmentation matches annotated boundaries. Although these measures are also helpful for analyzing the quality of object proposals (see Ch. 5 - Ch. 7), the problem in object proposal generation is generally different. First, object proposal generation systems typically produce hundreds or thousands of object proposals. To cope with this, only the proposal that best matches an annotated object is evaluated. For a fair comparison, the set of proposals considered for this match is truncated at certain levels (e.g., the first 10, 100, or 1000 proposals). Second, we want to assess two main qualities of object proposals. On the one hand, we want to measure how many annotated objects are discovered by the proposals. On the other hand, we also want

(a) IoU = 0.0 **(b)** IoU ≈ 0.1 **(c)** IoU ≈ 0.5 **(d)** IoU ≈ 0.5

(e) IoU ≈ 0.7 **(f)** IoU ≈ 0.8 **(g)** IoU ≈ 0.9 **(h)** IoU ≈ 0.9

Figure 2.11: Visualization of different *Intersection over Union* (IoU) levels between a proposal (red) and the annotated object (green). The intersection of the annotated object with the proposals is highlighted in blue. The proposals in (a) and (b) do not discover the object (IoU < 0.5). In contrast, the proposals in (c) and (d) roughly discover the entire object or parts of the object in more detail. The proposals in (e), (f), and (g) refine the proposal from (d), leading to higher IoU values. Note that the proposal in (g) almost perfectly discovers the object while achieving only an IoU of 0.9. Similarly, the proposal in (h), which misses both feet and a hand, also has an IoU of 0.9. Base image and annotation taken from the LVIS dataset [Gupta et al., 2019].

to assess how precisely the proposals segment the objects[6]. Note that precision[7] is generally not an important measure for object proposals [Zitnick and Dollár, 2014; Hosang et al., 2015], since the generation of proposals usually serve as a pre-processing step.

The qualities mentioned above, how many objects are discovered and how well they are segmented, are closely related. This relation becomes apparent when assessing which proposals in Fig. 2.11 discover the annotated object (tennis player). A common measure to decide if a proposal region $P \subseteq \Omega$ discovers an annotated object region $O \subseteq \Omega$ is the overlap assessed by the *Intersection over Union* (IoU), defined as

$$\mathrm{IoU}(P, O) = \frac{|P \cap O|}{|P \cup O|}. \tag{2.6}$$

The IoU calculates the overlap independent of the proposal's size and the object's size. Given the IoU, we can define the set of true positives TP_t for an image as the number of annotated objects discovered by at least one proposal with an IoU of at least t. Similarly, the set of false negatives FN_t is defined as the number of annotated objects that are not discovered by at least one proposal with an IoU of at least t. Overall, we define the *Recall* ($\mathrm{Rec}(t)$) given TP_t and FN_t as

$$\mathrm{Rec}(t) = \frac{\mathrm{TP}_t}{\mathrm{TP}_t + \mathrm{FN}_t}. \tag{2.7}$$

Everingham et al. [2010] suggest a threshold of $t = 0.5$ as a lower bound for a proposal to discover an object successfully. As visible from the images in Fig. 2.11, an IoU of 0.5 means that the object is only roughly discovered. Hence, many authors not only report the Rec(0.5)

[6] Assessing the segmentation is independent of the type of proposal (masks or boxes).
[7] Precision in this context assesses how many object proposals discover objects.

Table 2.1: Overview of different variations of the Average Recall (AR) w.r.t. the number of proposals analyzed and the size of the considered annotated objects.

Variation	Number of proposals	Size (a) of considered objects in pixels	Description
AR@10	10	all	AR for the first 10 proposals compared to annotated objects of all sizes.
AR@100	100	all	AR for the first 100 proposals compared to annotated objects of all sizes.
AR@1000/ AR@1k	1000	all	AR for the first 1000 proposals compared to annotated objects of all sizes.
AR^S@100	100	$a < 32^2$	AR for the first 100 proposals compared to **small** annotated objects.
AR^M@100	100	$32^2 \leq a \leq 96^2$	AR for the first 100 proposals compared to **medium** annotated objects.
AR^L@100	100	$96^2 < a$	AR for the first 100 proposals compared to **large** annotated objects.

but take the concept one step further and plot curves with t in the interval of $[0.5, 1]$ [Endres and Hoiem, 2010; Krähenbühl and Koltun, 2014; Zitnick and Dollár, 2014]. This allows a better analysis of how well the proposals adhere to actual the object boundaries. Hosang et al. [2015] propose to integrate $\text{Rec}(t)$ over the interval $[0.5, 1]$, which leads to one convenient number for comparison, given a fixed number of proposals. Thus, [Hosang et al., 2015] define the *Average Recall* (AR) as[8]

$$\text{AR} = 2 \int_{0.5}^{1} \text{Rec}(t)\, \mathrm{d}t. \tag{2.8}$$

Hosang et al. [2015] also show that the AR correlates with the results of object detection systems using these proposals. Overall, AR measures both qualities of object proposals: the number of discovered objects and the segmentation quality of the proposals.

AR is usually reported for 10, 100, and 1000 proposals to compare object proposal generation results for different numbers of proposals (AR@10, AR@100, and AR@1000). Moreover, size-specific ARs for the first 100 proposals (AR^S@100, AR^M@100, and AR^L@100) assess the quality of proposals w.r.t. the annotated objects of specific size ranges. Following Lin et al. [2014], small objects (S) have an absolute area of less than 32^2 pixels, while large objects (L) have an area of more than 96^2 pixels. The remaining objects are of medium size (M). Note that these absolute areas were defined for the images of the COCO dataset and may not be reasonable for datasets using other images. Similar to other authors [Pinheiro et al., 2015, 2016; Hu et al., 2017a], we report AR throughout this thesis since it covers both qualities of object proposals and is correlated with subsequent object detection results. Table 2.1 presents an overview of the different AR variations reported in this thesis.

Besides AR, the *Average Best Overlap* (ABO) [Endres and Hoiem, 2010; Hosang et al., 2015] was frequently used to evaluate object proposal generation systems [Uijlings et al., 2013;

[8] In practice, the integral is approximated by a sum at the steps $0.5, 0.55, 0.6, \ldots, 0.95$ for t.

Krähenbühl and Koltun, 2014; Pont-Tuset et al., 2017]. ABO measures the best overlap in terms of IoU between a set of proposals and an annotated object. Subsequently, the best overlap is averaged over all annotated objects in the image. However, Hosang et al. [2015] show that ABO is identical to AR if the best overlap for an annotated object is set to 0 in case it is below 0.5. Hence, we do not report ABO in this thesis.

Chapter 3

Related Work

Table of Contents

The contributions of this thesis focus on superpixel segmentation and object proposal generation. To better understand the context of the contributions, we review relevant literature in the fields of superpixel segmentation (see Sec. 3.1) and object proposal generation (see Sec. 3.2). We also discuss the strengths and limitations of several approaches. Some of these limitations were already mentioned in Sec.1.2 and will be addressed in the subsequent chapters of the thesis.

3.1 Literature on Superpixel Segmentation

This section discusses various approaches to generate superpixel segmentations on static 2D color or intensity images. The discussion only includes approaches that aim to generate superpixel segmentations or oversegmentations. Hence, we exclude early approaches on image segmentation [Haralick and Shapiro, 1985] or approaches on semantic segmentation [Chen et al., 2016a, 2017] and instance segmentation [Hariharan et al., 2014; He et al., 2017a].

Since Ren and Malik [2003] defined the term superpixel in 2003, many superpixel segmentation approaches were proposed. To give a structured overview, we follow the general taxonomy of Achanta et al. [2012] that divides superpixel segmentation methods into *gradient ascent-based* methods and *graph-based* methods. Gradient ascent-based methods start from an initial segmentation and subsequently improve this segmentation in one or multiple steps. Graph-based methods represent the image as a graph and apply cuts to the graph or merge parts of the graph to generate superpixels. Furthermore, we add the class of *CNN-based* methods to reflect recent advances in this area. Figure 3.1 visualizes this taxonomy with its

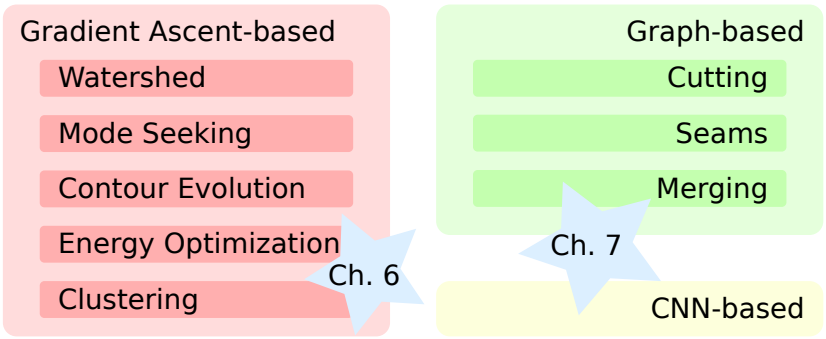


Figure 3.1: Overview of our taxonomy for superpixel segmentation methods with sub-categories inspired by Achanta et al. [2012] and Stutz et al. [2018]. The stars locate our contributions of Ch. 6 and Ch. 7 within the taxonomy. While the contribution in Ch. 6 adapts energy optimization and clustering methods, the contribution in Ch. 7 links graph-based merging and CNN-based methods.

Table 3.1: Overview of the superpixel segmentation methods most frequently utilized in this thesis: ETPS [Yao et al., 2015], SLIC [Achanta et al., 2012], FH [Felzenszwalb and Huttenlocher, 2004], SEAL [Tu et al., 2018], SSN [Jampani et al., 2018], and SpxFCN [Yang et al., 2020]. Oversegmentation denotes the amount of oversegmentation that the methods produce. Number of superpixels describes the ability of methods to produce a desired number of superpixels. Runtime indicates the average runtime on images from the BSD dataset based on Stutz et al. [2018]. The number of stars represents a rating from high runtime/oversegmentation (∗) to low runtime/oversegmentation (∗∗∗).

Method	Class	Oversegmentation	Number of superpixels	Runtime	Used in
ETPS	Gradient → Energy	∗	✓	∗∗	Ch. 5 & Ch. 6
SLIC	Gradient → Clustering	∗	✓	∗∗	Ch. 5 & Ch. 6
FH	Graph → Merging	∗∗∗	✗	∗∗	Ch. 5 & Ch. 7
SEAL	CNN	∗	✓	∗	Ch. 7
SSN	CNN	∗∗	✓	∗∗	Ch. 7
SpxFCN	CNN	∗	✗	∗∗∗	Ch. 7

sub-categories. The stars in Fig. 3.1 locate our contributions in Ch. 6 and Ch. 7 w.r.t. the taxonomy.

In the following three sections, we give a brief overview of the most relevant and influential approaches in the literature. Moreover, we will discuss fundamental approaches for this thesis in more detail. Table 3.1 presents an overview of these fundamental approaches with key properties and a classification w.r.t. our taxonomy. The presentation of the literature on superpixel segmentation methods closes with a discussion in Sec. 3.1.4.

3.1.1 Gradient Ascent-based Superpixel Segmentation

Gradient ascent-based methods for superpixel segmentation start from an initial segmentation that is refined in multiple steps. Following Stutz et al. [2018], we subdivide gradient ascent-based approaches into methods based on *watersheds*, *mode seeking*, *contour evolution*, *energy optimization*, and *clustering*. We will discuss each class in more detail in the subsequent sections.

Watersheds

The principle of watershed-based segmentation methods is inspired by topography and the flow of raindrops or water into catchment basins. In this analogy, the catchment basins are the final superpixels. Vincent and Soille [1991] and Meyer [1994] presented two classic formulations for oversegmentation based on watersheds before the emergence of superpixels. Vincent and Soille [1991] recursively enlarge catchment basins from their minima or markers. A dam is constructed as a watershed leading to a superpixel boundary whenever two catchment basins touch. Alternatively, Meyer [1994] assigns pixels to minima or markers based on the path of the steepest slope using topographical distance. All pixels assigned to one minimum or marker form a superpixel. A common problem when using the watershed principle for superpixel segmentation are non-compact superpixels [Levinshtein et al., 2009; Achanta et al., 2012]. To enforce compactness, different authors [Neubert and Protzel, 2014; Hu et al., 2015] propose compactness constraints based on the Euclidean distance between pixels and minima or markers.

Mode Seeking

Mode seeking-based superpixel segmentation methods interpret the image pixels as a Probability Density Function (PDF) in a feature space of a joint spatial and color domain [Comaniciu and Meer, 2002]. Within the PDF, the pixels are linked to modes that form clusters and ultimately lead to superpixels. Mode seeking-based superpixel segmentation methods start by estimating a density. Based on this density, each data point (pixel) moves towards the modes of the density function utilizing the gradient information. Comaniciu and Meer [2002] propose using mean shift [Fukunaga and Hostetler, 1975] for this mode seeking. In mean shift, each data point is updated until convergence by taking the weighted mean of its neighborhood and moving the point accordingly. Sheikh et al. [2007] extend the concept to medoid shifts by replacing the mean with the median. Improving medoid shift, Vedaldi and Soatto [2008] reduce the computational complexity and integrate a parameter for controlling the amount of over- or undersegmentation.

Contour Evolution

Superpixel segmentation methods based on contour evolution iteratively grow superpixels from seeds, similar to a diffusion process. The velocity at which superpixel contours evolve is based on a flow representing the image complexity. In uniform areas, the flow is high. In areas with edges, the flow is low. Levinshtein et al. [2009] start from regularly placed seeds and calculate the flow around seeds or contour pixels based on the gradient magnitude and the distance to other superpixels. This leads to inefficient processing and non-compact superpixels. Variations of the original concept exist [Wang et al., 2013a; Buyssens et al., 2014], which adapt the flow computation and include color homogeneity or compactness constraints.

Energy Optimization

Approaches that use an energy optimization framework to generate superpixel segmentations start from a regular grid of superpixels. Subsequently, they evolve this initial superpixel

segmentation by optimizing an energy function. One of the seminal approaches in this category is the *Superpixels Extracted via Energy-Driven Sampling* (SEEDS) [Van den Bergh et al., 2015]. The energy function in SEEDS favors superpixels of homogeneous color and regular shape. For optimization, a hill-climbing strategy is used that reassigns blocks of pixels or pixels from one superpixel to an adjacent one in a coarse to fine manner. This optimization refines the initial segmentation and leads to superpixels adhering to the image contents. However, the superpixels lack compactness and produce a substantial amount of oversegmentation.

Using a similar coarse to fine optimization, Yao et al. [2015] present the *Extended Topology Preserving Segmentation* (ETPS)[1] that extends the energy function of SEEDS. Besides homogeneous color and a regular shape per superpixel, the energy function also preserves the topology and enforces a minimum size. These additional constraints improve the results but also increase the amount of oversegmentation. We will discuss ETPS in more detail below since we modify ETPS in Ch. 6. Extending ETPS, Lee et al. [2017] add a term to the energy function that estimates the likelihood of a boundary between superpixels based on a codebook of boundary patches.

ETPS Borrowing the idea of SEEDS to utilize a hierarchical optimization, Yao et al. [2015] propose *Extended Topology Preserving Segmentation* (ETPS). The optimization in ETPS is based on an energy function similar to SEEDS. However, ETPS includes explicit terms to control the topology (no disconnected superpixels), the size (uniformly sized superpixels), and a second shape regularization term. Overall, the energy function E in ETPS consists of five parts given a superpixel segmentation $S = \{S_1, \dots S_n\}$:

$$E(S) = \sum_{S_i \in S} E_c(S_i) + \gamma_s \sum_{S_i \in S} E_s(S_i) + \gamma_b E_b(S) + E_t(S) + E_m(S) \tag{3.1}$$

with weights γ_p and γ_b to balance the influence of the different terms. E_t and E_m do not need weights, since they either evaluate to 0 or ∞ as described below. The first term (E_c) controls the color homogeneity of each superpixel by calculating the squared Euclidean distance between the mean color value of a superpixel $\boldsymbol{\mu}_{c,S_i}$ and the color value of each pixel $I_{\mathrm{RGB}}(\mathbf{p}) : \Omega \to \mathbb{R}^3$ within the superpixel. Thus, E_c is defined as

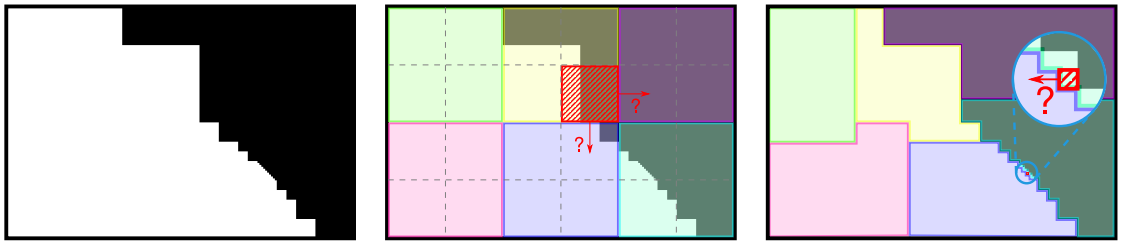
$$E_c(S_i) = \sum_{\mathbf{p} \in S_i} \| I_{\mathrm{RGB}}(\mathbf{p}) - \boldsymbol{\mu}_{c,S_i} \|_2^2. \tag{3.2}$$

The second term (E_s) controls the spatial extent by calculating the squared Euclidean distance between the center of a superpixel $\boldsymbol{\mu}_{s,S_i}$ and each pixel $\mathbf{p}$ within the superpixel:

$$E_s(S_i) = \sum_{\mathbf{p} \in S_i} \| \mathbf{p} - \boldsymbol{\mu}_{p,S_i} \|_2^2. \tag{3.3}$$

The term E_b controls the compactness of the superpixels by counting how many pixels have an 8-neighborhood consisting of pixels belonging to more than one superpixel. If any superpixel in S is disconnected, E_t imposes an infinite cost to preserve the topology. Similarly, E_m imposes infinite cost if any superpixel in S is smaller than a specific minimum size.

[1] We use the name proposed by Stutz et al. [2018].



(a) Input image **(b)** Coarse block-level reassignment **(c)** Fine pixel-level reassignment

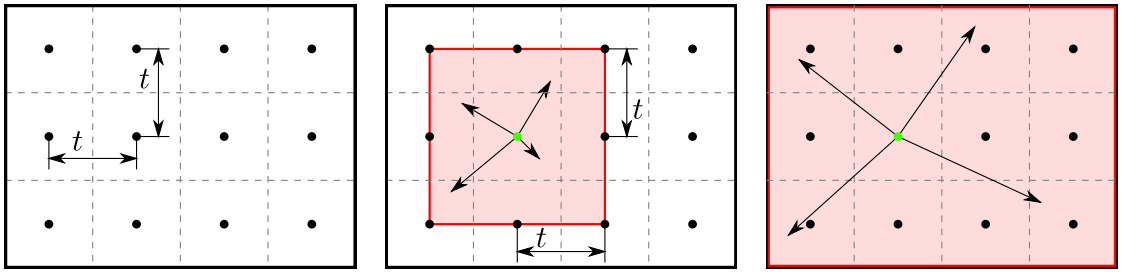
Figure 3.2: Example of the hierarchical optimization in ETPS for a simple input image (a) with six superpixels denoted by colors in (b) and (c). First, coarse block-level reassignments are considered, like reassigning the red block in (b) from the yellow superpixel to the violet or the blue superpixel. On the finest level, individual pixels are considered, like the red pixel highlighted in (c). The circled region in (c) is enlarged for visualization.

For optimizing the energy function E, ETPS enforces a hierarchical approach starting from a regular grid of equally sized superpixels. Following SEEDS, the hierarchical optimization reassigns blocks of pixels (see Fig.3.2(b)) or individual pixels (see Fig. 3.2(c)) from one superpixel to an adjacent one. Subsequently, the energy function is evaluated given the new assignment. If the reassignment reduces the energy, the assignment is kept. Different from SEEDS, ETPS utilizes a queue to reassign blocks along current superpixel boundaries systematically. Once the queue is empty, and all blocks are processed, the following hierarchy level with finer blocks is processed. After all hierarchy levels are processed, compact superpixels emerge that are uniformly distributed across the image, which supports oversegmentation.

Clustering

Clustering-based superpixel segmentation methods also start from an initial segmentation and iteratively refine the superpixels. Achanta et al. [2012] started one prominent line of work with their *Simple Linear Iterative Clustering* (SLIC). We will discuss SLIC below in more detail since we modify it in Ch. 6. In a nutshell, SLIC applies a k-means clustering to the image pixels in a 5D feature space composed of three color values and the spatial coordinates. Overall, a pre-defined number of superpixels with tunable compactness emerge. Since the superpixels are uniformly distributed across the image, SLIC superpixel segmentations exhibit a large amount of oversegmentation. Several works follow up on SLIC to improve the original formulation. Enriching the feature space, Li and Chen [2015] map the original 5D feature space into a 10D feature space using the Fourier series. Achanta and Süsstrunk [2017] improve the computational efficiency of SLIC by limiting the number of iterations to one utilizing an online updating of cluster centers. Enforcing smaller superpixels in image areas with details, Liu et al. [2016b] apply SLIC on a 2D manifold of the original SLIC features.

Apart from SLIC and its variations, several other clustering-based methods exist. Wang and Wang [2012] extend the centroidal Voronoi tessellation with color features and a compactness term to generate superpixel segmentations. Focusing on efficient processing, Shen et al. [2016] utilize the DBSCAN clustering method based on color, spatial coordinates, and a size constraint. Ban et al. [2018] use a Gaussian mixture model over spatial and color features to create superpixel segmentations.



(a) Initial seed placement **(b)** Local k-means **(c)** Original k-means

Figure 3.3: Initial placement of k seeds (black dots) in regular intervals of size t on an image in SLIC (a) as well as a comparison of the local k-means (b) and the original k-means clustering (c). While the local k-means utilized in SLIC has a restricted search region (red area) around a seed (green), the original k-means clustering considers the entire image as the search region. Figure adapted from Achanta et al. [2012].

SLIC *Simple Linear Iterative Clustering* (SLIC) proposed by Achanta et al. [2012] utilizes a local k-means clustering of pixels in a 5D feature space based on the color values and the spatial coordinates of pixels. SLIC starts by placing a pre-determined number of seeds in regular intervals on the image. The distance on the x- or y-axes between the k seed points is defined as

$$t = \sqrt{\frac{h \cdot w}{k}} \tag{3.4}$$

for an image with height h and width w (see Fig. 3.3(a)). To avoid seeds on edges, the seeds are moved to the pixel with the lowest gradient magnitude within the 8-neighborhood.

Based on these seeds as cluster centers, SLIC applies a local k-means clustering. For each seed or cluster center, the distance in the 5D feature space is computed w.r.t. all pixels with an L_1-distance of less or equal t on the image plane. The spatially restricted calculation speeds up the processing compared to the original k-means. Figure 3.3 visualizes a comparison with the original k-means, highlighting the smaller search region (red area). Subsequently, each pixel is assigned to the cluster center with the smallest distance. The cluster centers are updated in the 5D feature space by taking the mean value per feature as the original k-means. This procedure is repeated iteratively to adapt the initial superpixels to the image content. Termination is reached once the cluster centers have converged.

An essential part of SLIC is the 5D feature space. Given a pixel $\mathbf{p}$ from the set of image pixels Ω, the feature space consists of the LAB values from the CIELAB color space ($I_{\text{LAB}}(\mathbf{p}) : \Omega \to \mathbb{R}^3$) as well as the pixel's x- and y-coordinates ($\mathbf{p}$). Since the distances between the LAB values and the spatial distance are on different scales, the overall distance in the feature space is a weighted sum of two distances. The spatial distance δ_s of two pixels $\mathbf{p}_i, \mathbf{p}_j \in \Omega$ is the Euclidean distance:

$$\delta_s = \|\mathbf{p}_i - \mathbf{p}_j\|_2. \tag{3.5}$$

Similarly, the distance between the LAB color values of the two pixels $\mathbf{p}_i, \mathbf{p}_j \in \Omega$ is defined as the Euclidean distance between the vectors of color values:

$$\delta_c = \|I_{\text{LAB}}(\mathbf{p}_i) - I_{\text{LAB}}(\mathbf{p}_j)\|_2. \tag{3.6}$$

To balance the distances, normalization coefficients are used. δ_s is normalized using t since it approximately limits the spatial distance within each cluster. In contrast, no proper

normalization coefficient can be derived for δ_c, since the colors may vary significantly between superpixels and images. Therefore, a parameter m is introduced that normalizes δ_c and controls the importance of the spatial distance δ_s and the distance in color space δ_c. The overall distance δ in the 5D feature space used for clustering is defined as

$$\delta = \sqrt{\left(\frac{\delta_c}{m}\right)^2 + \left(\frac{\delta_s}{t}\right)^2}. \tag{3.7}$$

As a result, m allows the user to control the compactness of the superpixels.

After termination of the clustering, a post-processing eliminates disconnected superpixels by assigning them to one of their neighboring superpixels. Overall, SLIC efficiently generates uniformly distributed superpixels with good segmentation quality and a large amount of oversegmentation. Additionally, the user controls the number of generated superpixels (k) and the superpixels' compactness (m).

3.1.2 Graph-based Superpixel Segmentation

In contrast to gradient ascent-based approaches, which start from an initial superpixel segmentation or seeds, graph-based approaches generate components in an image graph to create superpixels. The image graph represents image pixels as nodes and the adjacency of pixels as edges. The edges are weighted, representing the similarity of the two linked pixels. Components (superpixels) are generated by cutting the graph, adding seams, or merging vertices. We will discuss these three types of graph-based superpixel segmentation approaches in more detail below.

Cutting

Graph cuts seek a partition of the image graph to generate a superpixel segmentation. Shi and Malik [2000] propose normalized cuts to generate this partition. Normalized cuts extend the minimum cut in graph theory, which minimizes the accumulated weight of the edges removed to create a cut in the graph. Shi and Malik [2000] add a normalization term to this weight accumulation to prevent a strong size imbalance between the superpixels generated by one cut. To create a pre-defined number of superpixels, normalized cut is applied recursively. Despite the simple idea behind normalized cuts, the runtime is high compared to other superpixel segmentation methods [Stutz et al., 2018].

Veksler et al. [2010] formulate energy functions over the pixel graph to assign each node/pixel to one of four half-overlapping patches. Subsequently, they utilize an efficient graph cut-based framework to optimize the assignment. However, the method is still slow compared to other superpixel segmentation methods [Stutz et al., 2018]. To increase the computational efficiency, Zhang et al. [2011a] replace the four half-overlapping patches per pixel with half-overlapping horizontal or vertical stripes to create a two-class problem. After solving the problem for horizontal and vertical stripes, the results are overlayed, and the final superpixels emerge.

Seams

Seam-based superpixel segmentation methods adapt the idea of seam carving [Avidan and Shamir, 2007]. Hence, those methods identify paths (seams) from top to bottom or left to right in the image that split the image. Moore et al. [2008] follow this principle and create superpixels with a regular grid topology similar to the image pixels. To keep the topology, Moore et al. [2008] seek non-overlapping horizontal or vertical seams through the image graph that follow the strongest edges. Subsequently, the final superpixel segmentation is created from overlaying the horizontal and vertical seams. Moore et al. [2010] reduce this formulation by creating only one seam in a multi-layer graph representing the image. For keeping the topology, constraints are encoded similar to Moore et al. [2008]. Overall, the seam-based approaches focus on preserving the regular grid topology, which impairs the segmentation quality compared to other modern approaches.

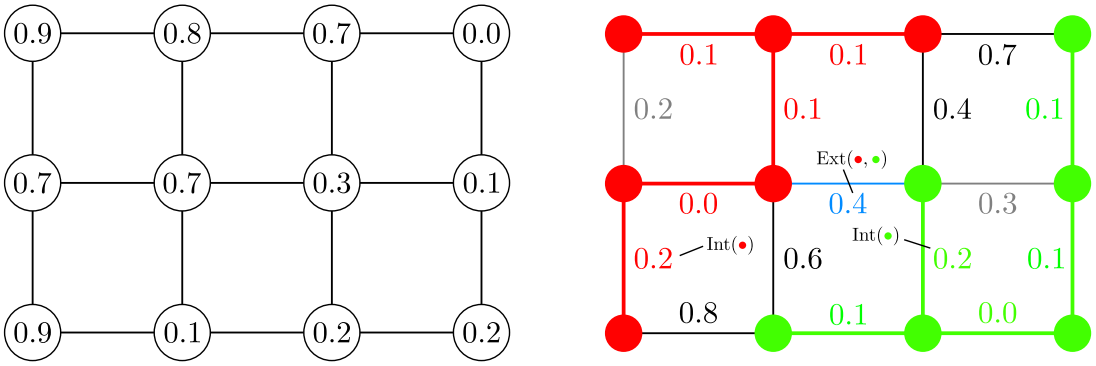
Merging

Another sub-class of graph-based approaches merge vertices (pixels) of the image graph to create superpixels. Felzenszwalb and Huttenlocher [2004] proposed the most influential approach in this category. The *Felzenszwalb and Huttenlocher* (FH) superpixel segmentation method utilizes the difference in intensity or color between pairs of adjacent pixels as edge weights. Subsequently, the vertices (pixels) or groups of vertices are merged until a termination criterion is reached. A more detailed explanation is given below since FH is utilized and extended in Ch. 5 and Ch. 7. Luengo et al. [2016] combine the concept of FH with minimum cuts in an iterative framework alternating merging and cutting of superpixels.

Another line of work on merging-based superpixel segmentation methods applies random walks on the image graph. Liu et al. [2011] use the entropy of random walks to create *Entropy Rate Superpixel Segmentation*s (ERS), leading to compact superpixels of uniform size. Shen et al. [2014] use random walks to estimate a pixels' association to pre-defined seeds. After assigning each pixel to a seed based on the shortest random walk, the seeds are relocated, non-uniform superpixels are split, and a new iteration starts.

FH Felzenszwalb and Huttenlocher [2004] proposed the *Felzenszwalb and Huttenlocher* (FH) superpixel segmentation method, which greedily merges vertices (pixels) in the image graph. Reaching this goal, FH starts by setting up the image graph $G = (V, E)$ with a vertex $v_i \in V$ for each pixel and an edge $(v_i, v_j) \in E$ linking the vertices that represent adjacent pixels. Additionally, each edge $e_i \in E$ is weighted ($w(e_i) : E \rightarrow \mathbb{R}^+$) based on the intensity or color difference of the linked pixels. To control the merging of the components (pixels or groups of pixels), Felzenszwalb and Huttenlocher [2004] use the fundamental properties of superpixels. First, a superpixel should have a low internal difference (homogeneous). Second, adjacent superpixels should have a substantial external difference (cover different things). The internal difference of a component $C \subset V$ in FH is the largest edge weight within the minimal spanning tree (MST) of C (see Fig. 3.4(b) for an example):

$$\mathrm{Int}(C) = \max_{e \in \mathrm{MST}(C)} w(e). \tag{3.8}$$



(a) Image graph **(b)** Intermediate step with two components

Figure 3.4: Example of an iteration in the merging process of FH with two components denoted with red and green (b). The 12 vertices in the image graph (a) represent the 12 image pixels with their respective pixel intensities, while the values along the edges in (b) denote the intensity differences. Given the components and edge weights, FH updates the minimal spanning trees of the two components marked by the red and green edges. Subsequently, the smallest remaining edge not assigned to a component is determined (blue). Since this edge connects the red and the green components, the internal difference (Int) for both components and the external difference (Ext) between the components are computed. Given Eq. 3.10 with $k = 1$, the components are not merged, since the external difference ($\mathrm{Ext}(\bullet, \bullet) = 0.4$) is larger than the combination of the internal differences ($\mathrm{MInt}(\bullet, \bullet) = 0.37$).

For assessing the external difference between two components $C_1, C_2 \subset V$, FH uses the smallest weight of any edge connecting C_1 and C_2 (blue edge in Fig. 3.4(b)):

$$\mathrm{Ext}(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j)). \tag{3.9}$$

Given both internal differences and the external difference of two components C_1 and C_2, FH combines them into one condition $\mathrm{D}(C_1, C_2)$ to control the merging (true means components are merged):

$$\mathrm{D}(C_1, C_2) = \begin{cases} \text{false} & \mathrm{Ext}(C_1, C_2) > \mathrm{MInt}(C_1, C_2), \\ \text{true} & \text{else.} \end{cases} \tag{3.10}$$

$\mathrm{MInt}(C_1, C_2)$ combines the internal differences of the components utilizing the parameter k:

$$\mathrm{MInt}(C_1, C_2) = \min\left(\mathrm{Int}(C_1) + \frac{k}{|C_1|}, \mathrm{Int}(C_2) + \frac{k}{|C_2|}\right). \tag{3.11}$$

Given the two internal differences, MInt only considers the smaller one and adds a constant k normalized by the cardinality of the respective component. Thus, if the components are small, the offset is larger, allowing more dissimilar components to merge early. Once the components grow, merging takes only place if the components are almost uniform.

The merging process itself is carried out greedily. FH sorts all edges in G in ascending order given their weights and iteratively evaluates the condition D for each edge. If D evaluates to true, the components linked by the edge are merged into one component. Ultimately, each component in G represents one superpixel. Felzenszwalb and Huttenlocher [2004] show that the resulting superpixel segmentation is neither too fine nor too coarse, given the condition D. Additionally, the order of edges in merging is irrelevant as long as it is non-decreasing.

Since neither component of FH includes a size or compactness constraint, the size and compactness of the superpixel might vary. This lack of compactness or size constraints results in a reduced oversegmentation compared to other methods. Moreover, there is only indirect control over the number of superpixels by varying the only parameter k, which controls the maximum difference between the internal and external differences. Therefore, the number of superpixels primarily depends on the image content.

3.1.3 CNN-based Superpixel Segmentation

As discussed in the introduction, few CNN-based approaches on superpixel segmentation were published. The approaches generally follow one of three directions. Tu et al. [2018] and Jampani et al. [2018] introduce CNN-based extensions of the traditional superpixel segmentation methods ERS [Liu et al., 2011] and SLIC [Achanta et al., 2012]. In contrast, Yang et al. [2020] and Wang et al. [2021c] approach the generation of superpixel segmentations as a per-pixel classification task. In this formulation, a CNN assigns each pixel to one of the initial superpixels. Using a different formulation, Zhu et al. [2021] extract CNN-based feature embeddings for clustering pixels in an unsupervised framework. Since we propose a CNN-based superpixel segmentation method in Ch. 7, we present the closely related works of Tu et al. [2018], Jampani et al. [2018], and Yang et al. [2020] in more detail below.

SEAL Tu et al. [2018] propose the system SEAL (*Segmentation-Aware Loss*), which extends the graph-based approach by Liu et al. [2011] utilizing a CNN. Liu et al. [2011] use the entropy of random walks on the image graph for creating superpixels. The edge weights in the image graph represent the difference between pixels in terms of intensity values. SEAL learns these weights as affinities using a simple CNN with ten convolutional layers given the image and the result of an edge detector. Based on the learned edge weights, SEAL applies the framework of Liu et al. [2011] to generate superpixels.

The major contribution of Tu et al. [2018] is learning the edge weights as affinities. Tu et al. [2018] argue that using binary cross-entropy loss for learning the affinities does not fit the superpixel segmentation task. For instance, a single misclassified pixel can lead to severe undersegmentation errors. To prevent this, SEAL incorporates the superpixel segmentation result using the current affinities into the training by calculating an undersegmentation error for each superpixel as a weight. The weight emphasizes pixels along the annotated boundaries that lead to a high undersegmentation error and is used as in the binary cross-entropy loss. Hence, the error is calculated w.r.t. the superpixel segmentation task. Overall, the superpixels share the characteristics of Liu et al. [2011] but adhere better to the object boundaries.

SSN Similar to SEAL, the *Superpixel Sampling Networks* (SSN) [Jampani et al., 2018] is a CNN-based variation of a traditional superpixel segmentation method. Jampani et al. [2018] adapt SLIC [Achanta et al., 2012] by transferring it into a fully differentiable version that allows end-to-end training. Since SLIC applies a local k-means algorithm, it is not differentiable due to the hard association of one pixel to exactly one cluster. SSN overcomes this by using soft associations between each pixel and the cluster centers. To increase efficiency, the computations are limited to the surrounding 9 cluster centers/superpixels for each pixel, similar to the original formulation in SLIC. A hard relation is derived from the soft associations to create the final superpixels.

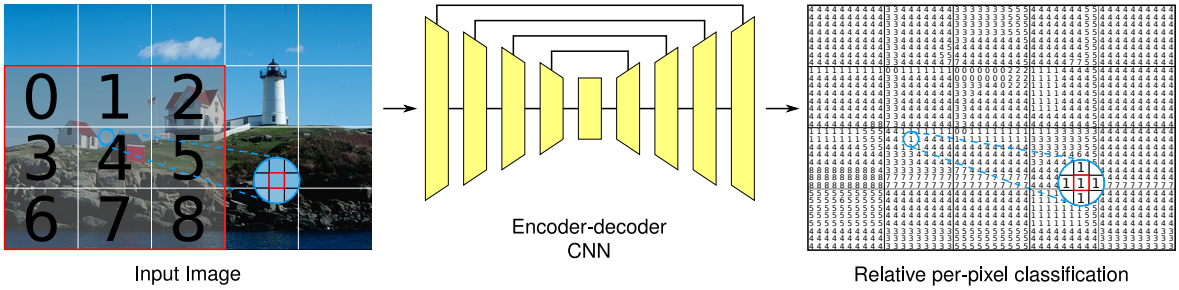


Figure 3.5: Overview of the system SpxFCN. SpxFCN assigns each pixel to one of the 9 initial superpixels around it. In the example on the left, the highlighted pixel is assigned to one of the numbered superpixels. Note that the numbers are relative to the location of the classified pixel. Therefore, the classification result (right) of the encoder-decoder CNN (middle) is always in the range $0, 1, 2, \ldots, 8$ indicating the relative assignment. Hence, the highlighted pixel is assigned to the upper superpixel (1). Input image taken from the BSD dataset [Martin et al., 2001].

The differentiable character of SSN allows Jampani et al. [2018] to augment the LAB color features with end-to-end learned CNN-based features. To this end, SSN uses a CNN for feature extraction that is linked with the differentiable SLIC to provide the learned input features. The feature extractor has a lightweight design with only seven convolutional layers. The loss function of SSN combines cross-entropy and a compactness loss that enforces spatially compact superpixels. Overall, SSN generates superpixels utilizing deep features in an end-to-end framework to increase the segmentation quality. Despite a strong methodological similarity with SLIC, the superpixels have a slightly different structure with less compactness and a lower amount of oversegmentation.

SpxFCN Yang et al. [2020] propose an end-to-end trainable CNN for superpixel segmentation (SpxFCN). Unlike SEAL or SSN, SpxFCN does not adapt an existing superpixel segmentation method. Instead, SpxFCN assigns each pixel to an initial superpixel grid cell in a classification framework. These superpixel grid cells are similar to the initial setup in superpixel segmentation methods like SLIC [Achanta et al., 2012]. To increase efficiency, each pixel is only assigned to one of the nine closest superpixels similar to SSN. Figure 3.5 visualizes the basic classification principle. To determine the classification result, SpxFCN consists of a standard encoder-decoder architecture with ten convolutional layers in the encoder and skip connections between the encoder and the decoder.

To train SpxFCN, Yang et al. [2020] propose a loss function that enforces proximate pixels with the same user-defined property (e.g., color or semantic label) to be assigned to the same superpixel. Overall, SpxFCN presents an end-to-end trainable superpixel segmentation network. Similar to the other CNN-based superpixel generation networks, the generated superpixels are not as compact as for methods like SLIC. However, the adherence to the annotated boundaries is better as Yang et al. [2020] show.

3.1.4 Discussion

This section presented superpixel segmentation methods using various approaches. Gradient ascent-based methods start from an initial segmentation or seeds that mostly result in compact, uniformly distributed superpixels with substantial oversegmentation. In contrast, the majority of graph-based methods produce less oversegmentation. Due to the construction of most

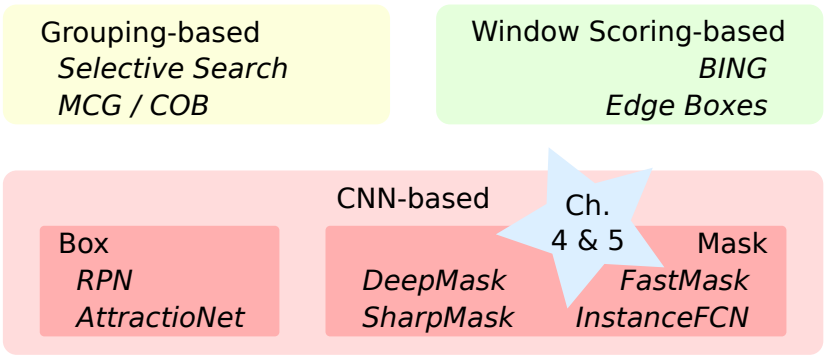


Figure 3.6: Overview of our taxonomy for object proposal generation systems based on Hosang et al. [2015]. The star locates our contributions of Ch. 4 and Ch. 5 within the taxonomy. Both chapters present contributions in the area of CNN-based object proposal generation with masks.

methods, only the intensity or color-based cut or merging criteria are relevant [Shi and Malik, 2000; Felzenszwalb and Huttenlocher, 2004], while a uniform spatial distribution is not enforced. The third class includes CNN-based methods that learn features to generate superpixel segmentations. The approaches either augment existing methods with CNNs [Tu et al., 2018; Jampani et al., 2018] or propose novel concepts [Yang et al., 2020; Wang et al., 2021c; Zhu et al., 2021]. Due to the construction, most CNN-based superpixel segmentation methods still exhibit a substantial amount of oversegmentation.

Overall, there exists a large body of literature on superpixel segmentation methods. Many approaches perform well on benchmark datasets but produce a large amount of oversegmentation (e.g., SLIC, ETPS, and SSN). In contrast, most methods that limit the oversegmentation suffer from inferior results on benchmark datasets (e.g., FH). However, such methods are essential for object proposal generation as discussed in the introduction. Hence, methods are missing that combine high segmentation quality with low oversegmentation. We will bridge this gap with our contributions in Ch. 6 and Ch. 7.

3.2 Literature on Object Proposal Generation

This section reviews the major object proposal generation systems proposed since the seminal work of Alexe et al. [2010]. Hosang et al. [2015] presented the first overview of the young object proposal generation field and proposed a taxonomy for object proposal generation systems to simplify the discussion. The taxonomy discriminates traditional approaches generating an objectness score for a set of windows (*window scoring-based*) and approaches grouping pixels or superpixels to form object proposals (*grouping-based*). Furthermore, we add the class of *CNN-based* object proposal generation systems to the taxonomy, with sub-classes for box-based and mask-based proposals. Figure 3.6 presents our taxonomy and some of the important methods. The star in Fig. 3.6 locates our contributions in Ch. 4 and Ch. 5 within our taxonomy.

The following sections discuss the most important approaches for each of the classes in the taxonomy. The discussion will focus on mask-based proposals and CNN-based approaches, as they are most related to our contributions in Ch. 4 and Ch. 5. Finally, Sec. 3.2.4 will summarize and discuss the literature on object proposal generation methods. Note that as object proposal generation is class-agnostic, we do not discuss approaches from related fields like object

detection or instance segmentation. Additionally, we only discuss approaches focusing on static 2D images and omit early methods [Russell et al., 2006; Malisiewicz and Efros, 2007; Gu et al., 2009] that utilize individual superpixels or all possible small groups of superpixels as proposals. These methods barely serve as baselines nowadays.

3.2.1 Window Scoring-based Object Proposal Generation

Window scoring-based object proposal generation systems consider a large set of windows (usually thousands of windows) and efficiently generate an objectness score per window. In their seminal works [Alexe et al., 2010, 2012], Alexe et al. use multiscale saliency to extract 100,000 windows at promising locations. They generate an objectness score per window, integrating the window's color contrast as well as edge, superpixel, and saliency information in a Bayesian framework. Chang et al. [2011] extend the approach of Alexe et al. [2010] by jointly learning per superpixel saliency and per window objectness. The joint learning leads to better object proposals since the tasks are closely related [Chang et al., 2011].

Rahtu et al. [2011] extract windows based on superpixel segmentations and a learned prior. For regressing objectness scores, they utilize superpixel boundary information and symmetry. Similarly, Cheng et al. [2014] propose their efficient system BING (*binarized gradient magnitude information* based on a learned strategy to extract windows and gradient information. However, as Zhao et al. [2014] show, the success of BING is mainly due to a sophisticated window extraction and pruning strategy rather than the gradient information. Zhang et al. [2011b] use an efficient cascade of classifiers to propose windows of different sizes and aspect ratios based on gradient features. To generate objectness scores, Zhang et al. [2011b] utilize edge features.

Using learned edge detection results, the system *Edge Boxes* [Dollár and Zitnick, 2013, 2014] efficiently evaluates up to millions of initial windows per image. Subsequently, Edge Boxes groups edges based on proximity and orientation to form contours. If several contours are located entirely within a window, Edge Boxes assigns a high objectness score to the window and vice versa. Concentrating the window extraction in Edge Boxes to the most interesting and diverse set of windows, Sun and Batra [2015] propose a branch and bound strategy. Taking a similar approach to Edge Boxes, Ma et al. [2017] use occlusion edges and compare the strength of edges crossing the boundaries of a window with the strength of edges close to but inside the boundaries of a window.

3.2.2 Grouping-based Object Proposal Generation

Grouping-based object proposal generation methods combine superpixels or pixels to form object proposals. As a result, the proposals are segmentation masks, although most authors still evaluate based on boxes. We follow Hosang et al. [2015] and discuss two different variations. First, we present grouping-based object proposal generation systems, which group superpixels with different strategies and features. Second, we review systems using graph cuts on pixel or superpixel graphs.

Superpixel Grouping

An early approach that groups superpixels to generate object proposals is *Selective Search* [van de Sande et al., 2011]. Starting from an initial superpixel segmentation [Felzenszwalb and Huttenlocher, 2004], Selective Search merges superpixels or superpixel groups greedily, yielding a hierarchy of superpixel segmentations. Merging is executed based on SIFT-like features [Lowe, 2004] and size to force small superpixels to merge early. All superpixels across the stages of the hierarchy comprise the set of proposals. To diversify the proposals further, the initial superpixel segmentation is generated on multiple color spaces and with different parameters. Uijlings et al. [2013] enhance Selective Search with additional color and shape features as well as more color spaces. Several other extensions of Selective Search were presented. While Yanulevskaya et al. [2014] add more features and learn a classifier for merging, Wang et al. [2015] propose a multi-branch merging strategy with complementary classifiers. Extending the concept of Selective Search further, Xiao et al. [2015] propose a new adaptive distance metric for feature comparison. Since Selective Search lacks an objectness score, Lu et al. [2015] introduce a score based on closed contours.

Various other approaches based on grouping superpixels exist. Arbeláez et al. [2014] propose *Multiscale Combinatorial Grouping* (MCG). MCG merges superpixels across hierarchical superpixel segmentations, resulting in object proposals. We will discuss this approach in more detail below since it generates very competitive mask-based object proposals. Extending MCG, Maninis et al. [2016] propose *Convolutional Oriented Boundaries* (COB), generating object proposals with the same framework as MCG. However, the hierarchy of superpixel segmentations is generated utilizing CNN-based boundary detection results. We classify COB as a grouping-based approach since the proposal generation is conducted without deep learning (see MCG). Using one superpixel graph rather than a hierarchy, Manén et al. [2013] generate proposals based on minimum spanning trees employing Prim's algorithm. To diversify the proposal set and not cover the entire image, the initial vertex and the termination are randomized. Krähenbühl and Koltun [2014] use the edge strength between superpixels and a geodesic distance in the superpixel graph to construct object proposals as level sets between foreground and background seeds. For selecting foreground and background seeds, a classifier based on several features is proposed. In contrast, Lauri and Frintrop [2017] use a statistical approach to sample object proposals from a superpixel segmentation based on the distance dependent Chinese restaurant process.

Another line of grouping-based object proposal generation systems strongly relies on saliency. Frintrop et al. [2014] and Martín García et al. [2015] utilize saliency to locate objects and fuse superpixels in highly salient areas to form object proposals. For ranking the proposals, saliency and shape information are utilized. Werner et al. [2015] also use saliency to guide the initial seed selection in a framework similar to Manén et al. [2013] and rank the proposals based on Gestalt principles [Wertheimer, 1922].

MCG *Multiscale Combinatorial Grouping* (MCG) [Arbeláez et al., 2014; Pont-Tuset et al., 2017] generates object proposals based on hierarchies of superpixel segmentations. Each superpixel segmentation hierarchy is generated in an efficient normalized cut framework utilizing low- and mid-level features. The hierarchy represents the different strengths of edges between superpixels as depth. Hence, two superpixels separated in the upper hierarchy are more likely to represent different ground truth regions. To cover objects of all sizes, MCG uses three scale-specific superpixel segmentation hierarchies, which are combined into a fourth hierarchy.

Since the hierarchies of superpixel segmentations do not cover the objects directly, MCG uses a combinatorial grouping approach to generate object proposals.

The combinatorial grouping considers all possible proposals based on combinations of n superpixels across all levels of each superpixel segmentation hierarchy. The depth in the hierarchy initially ranks the proposals. Hence, proposals consisting of superpixels with strong contours are preferred. This combinatorial grouping is applied to all four superpixel segmentations hierarchies with $n \in \{1, 2, 3, 4\}$ leading to 16 pools of proposals. Given the 16 pools, MCG learns to select the proposals across the pools using a Pareto front optimization scheme. Finally, all selected proposals are ranked using simple size, shape, and contour features like compactness and aspect ratio in a learned linear regression framework. Due to the large number of possible combinations during grouping, MCG lacks computational efficiency while generating precise object proposals [Hosang et al., 2015].

Graph Cuts

Graph cut-based methods for object proposal generation apply the idea of graph cuts on pixel or superpixel level. Carreira and Sminchisescu [2010, 2011] proposed the first approach for object proposal generation using graph cuts. To diversify the proposals, they use a parametric minimum cut formulation on pixel level, which calculates cuts with various parameter values. Multiple foreground seeds are used for the minimum cut, while pixels along the image border always represent the background. Endres and Hoiem [2010, 2013] use a parametric minimum cut on the superpixel graph and capture long range dependencies between superpixels. Several other authors also use the parametric minimum cut framework on the superpixel graph. Humayun et al. [2014] increase the efficiency by sharing computation between the minimum cut solutions of different seeds, Lee et al. [2014] utilize mid-level features like contour closure and symmetry, and Humayun et al. [2015] enforce more object proposals of medium size.

Combining global and local minimum cuts on the superpixel graph, Krähenbühl and Koltun [2015] generate object proposals at different granularities. While global minimum cuts focus on prominent objects, local minimum cuts are applied to parts of the graph based on a location cue. Also considering global and local information, Rantalankila et al. [2014] mix Selective Search with parametric minimum cuts. Based on the superpixel hierarchy in Selective Search, they choose an intermediate level as the base level for further processing. From this level, superpixel merging based on local decisions similar to Selective Search and parametric minimum cuts based on global decisions are applied in parallel to generate a diverse set of proposals.

3.2.3 CNN-based Object Proposal Generation

Unlike the previously discussed approaches that use hand-crafted features, the CNN-based object proposal generation systems learn features and the generation of object proposals in an end-to-end framework. In the following, we give an overview of box-based and mask-based object proposal generation systems utilizing CNNs.

Box Proposals

One of the first CNN-based object proposal generation systems was proposed by Erhan et al. [2014], which use an AlexNet [Krizhevsky et al., 2012] to generate a fixed number of proposals. The proposal box coordinates and the objectness scores are directly inferred from the backbone network without window extraction employing fully connected layers. During training, Erhan et al. [2014] assign annotated objects to the generated proposals dynamically as part of the loss calculation, leading to an end-to-end trainable system. Other early systems only generate an objectness score based on CNNs for numerous windows [Ghodrati et al., 2015, 2017] or Edge Boxes proposals [Kuo et al., 2015; Novotny and Matas, 2015] without a CNN-based bounding box regression.

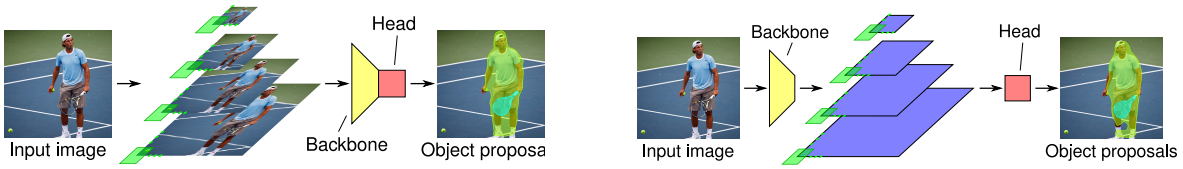
Extending the work of Erhan et al. [2014], Ren et al. [2016] propose *Region Proposal Network* (RPN). The RPN is integrated into the *Fast R-CNN* object detector [Girshick, 2015] and enables learning object proposal generation and subsequent object detection in one framework. To generate an arbitrary number of proposals, the RPN uses a sliding window approach with windows of different sizes and aspect ratios (anchor boxes) on top of a backbone network. The RPN regresses a box from each window and generates an objectness score using fully connected layers. Several extensions to the original formulation have been proposed. For instance, Kong et al. [2016] use features from multiple layers of the backbone to enhance the localization ability, while Lu et al. [2018] and Wang et al. [2019] learn the parameters of the anchor boxes with location, size, or aspect ratio priors for improved results.

In contrast to the previous approaches, Gidaris and Komodakis [2016] introduce *AttractioNet* and propose an iterative scheme to refine anchor boxes. Instead of directly regressing the box coordinates like Erhan et al. [2014] or Ren et al. [2016], AttractioNet regresses probability vectors for each box covering the box and a search area around it. The probability vectors represent the probability of each row or column in the search area being part of the object. By iteratively applying this procedure, initial coarse anchor boxes can adapt their position, size, and aspect ratio to the image content, minimizing the influence of the initial boxes.

Mask Proposals

For mask-based object proposals, few CNN-based systems exist. Generally, they follow either the *multi-shot concept* or the *one-shot concept* [Hu et al., 2017a]. Systems using the multi-shot concept extract windows from an image pyramid as visualized in Fig. 3.7(a). Subsequently, an objectness score and a segmentation mask are generated for each window utilizing a CNN like in *DeepMask* [Pinheiro et al., 2015] and *SharpMask* [Pinheiro et al., 2016]. DeepMask uses a backbone network and adds two heads for segmentation and objectness scoring, while SharpMask extends the concept by refining the initial coarse proposals. We discuss both systems in more detail below. Qiao et al. [2017] also use the multi-shot concept and extend SharpMask with a small network to learn application-specific downsampling factors for the image pyramid.

Applying a system on overlapping windows across different image pyramid levels like in the multi-shot concept is redundant and computationally inefficient. Therefore, Hu et al. [2017a] propose the *one-shot concept* for object proposal generation. In the one-shot concept, the backbone network processes an image only once. The construction of the pyramid is moved to the feature space as visualized in Fig. 3.7(b). Since the extraction of windows is conducted



(a) Multi-shot object proposal generation **(b)** One-shot object proposal generation

Figure 3.7: Comparison of multi-shot object proposal generation (a) and one-shot object proposal generation (b) according to Hu et al. [2017a]. In the multi-shot concept overlapping windows (green) from an image pyramid are extracted and processed individually by the backbone network (yellow) and the head (red) for segmentation and objectness scoring. In contrast, the image is only processed once by the backbone network (yellow) in the one-shot concept, yielding a feature pyramid (blue). From this pyramid, windows (green) are extracted and individually processed by the head (red). Input image taken from the COCO dataset [Lin et al., 2014]

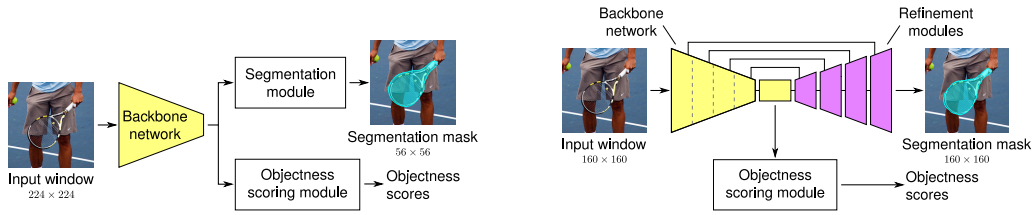
in feature space, the features generated by the backbone network are shared between all windows. Hu et al. [2017a] introduce *FastMask*, the first object proposal generation system following this concept. We present FastMask in more detail below since it serves as the base for our system presented in Ch. 4.

Dai et al. [2016] propose the hybrid system *InstanceFCN*, which incorporates ideas from both concepts and is inspired by semantic segmentation systems. InstanceFCN has two branches on top of a backbone network that processes the entire image at multiple resolutions like in the multi-shot concept. One branch generates an objectness score per pixel, while the other classifies the relative position[2] of each pixel w.r.t. the object center. For instance, an object pixel left of the object center is classified as left. To generate proposals, InstanceFCN extracts and evaluates several windows on the relative position prediction in a one-shot manner. Each window comprises nine sub-cells representing the nine relative positions. If a substantial amount of predictions match the relative position represented by the overlapping sub-cells, InstanceFCN generates a proposal from the locations of the matching predictions. Hence, InstanceFCN processes an image multiple times in the backbone like the multi-shot approaches but extracts windows in feature space like one-shot approaches.

DeepMask Pinheiro et al. [2015] presented the first CNN-based system for mask-based object proposal generation called *DeepMask*. DeepMask generates an image pyramid with rescaled versions of the input image and applies a CNN in a sliding window fashion following the multi-shot concept. The CNN consists of a backbone network, VGG-11 [Simonyan and Zisserman, 2015] in the original definition by Pinheiro et al. [2015], with two branches on top of the backbone as visualized in Fig. 3.8(a). The first branch creates the segmentation mask based on a convolutional layer and a fully connected layer by assigning the pixels to the classes object and non-object. All pixels belonging to the class object constitute the object proposal, which is generated on a coarse spatial resolution[3] (56×56) given features with an even lower spatial resolution (14×14). In parallel, based on the final feature map of the backbone, a second branch generates the objectness score for the input window.

[2]The relative positions are discretized in center, left, top-left, top, top-right, right, bottom-right, bottom, and bottom-left.

[3]The input window is of size 224×224.



(a) DeepMask **(b)** SharpMask

Figure 3.8: System overview of DeepMask [Pinheiro et al., 2015] (a) and SharpMask [Pinheiro et al., 2016] (b). Given an input image window, DeepMask generates a segmentation and an objectness score from the final feature map of the backbone network (yellow). Due to the reduced output size (56×56) compared to the input (224×224), the proposals are coarse and do not adhere well to the object boundaries. SharpMask uses the same backbone (yellow) and a similar processing for the objectness score. However, for generating the segmentation mask, refinement modules (pink) are introduced that gradually upsample the coarse features with features from earlier layers. As a result, the final segmentation mask has the same spatial resolution as the input window and captures more details of the objects. Input image window taken from the COCO dataset [Lin et al., 2014].

During training, only windows that wholly contain a centered object are considered positive samples. An object is centered if the center of the object is within 16 pixels of window's center. To be regarded as wholly contained, the object's longer dimension has to be between 48% and 68% of the respective window dimension. If an object does not fulfill these conditions, it is a negative sample. To train the mask generation branch, only positive samples are utilized since negative samples would only consist of single class entries (background). For the objectness branch, both positive and negative samples are used. The overall loss function combines binary logistic regression losses for the objectness score and the per-pixel classifier in the mask prediction branch. Note that the loss for the per-pixel classifiers is normalized by the number of pixels in the predicted segmentation.

The image pyramid in DeepMask consists of seven layers with downsampling factors ranging from 2^{-2} to 2^{1}. From each level, windows of size 224×224 with a stride of 16 pixels are extracted. Apart from this model, which we refer to as DeepMask, Pinheiro et al. [2016] defined the DeepMaskZoom model with an additional pyramid level $2^{-2.5}$. For fair comparison and better results, all DeepMask models in this thesis utilize a ResNet-50 backbone without the *conv5* stage (see Appendix A.2).

SharpMask The main drawback of DeepMask are the coarse segmentation masks. The downsampling process in the backbone network leads to a feature map of size 14×14 given an input window of size 224×224. Based on this 14×14 feature map, the 224×224 segmentation mask for the proposal is generated, which leads to a coarse segmentation mask due to interpolation. *SharpMask* [Pinheiro et al., 2016] utilizes a guided upsampling of the coarse features to segment proposals on input window resolution. This process is visualized in Fig. 3.8(b) and replaces the segmentation branch of DeepMask.

The guided upsampling employs refinement modules (pink blocks in Fig. 3.8(b)) that concatenate the upsampled, semantically rich features from the deeper layer with the features from the previous layer that have a higher spatial resolution. Four refinement modules are applied successively to recover features at input window resolution, utilizing different features from the backbone. Based on the final feature map, the segmentation mask is created with pixel-wise classifiers similar to DeepMask. SharpMask only refines proposals with a high

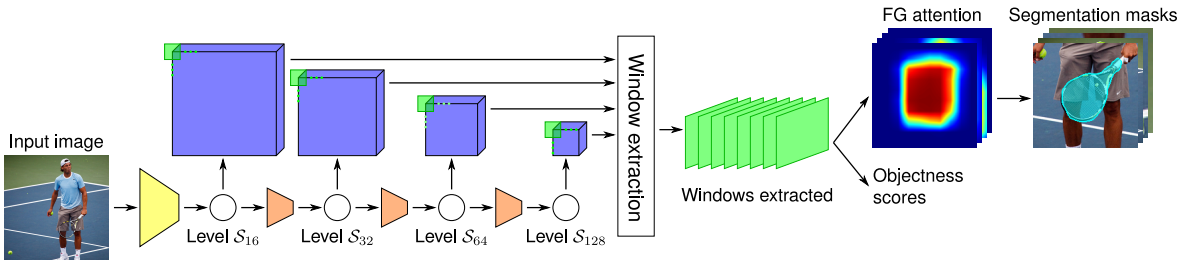


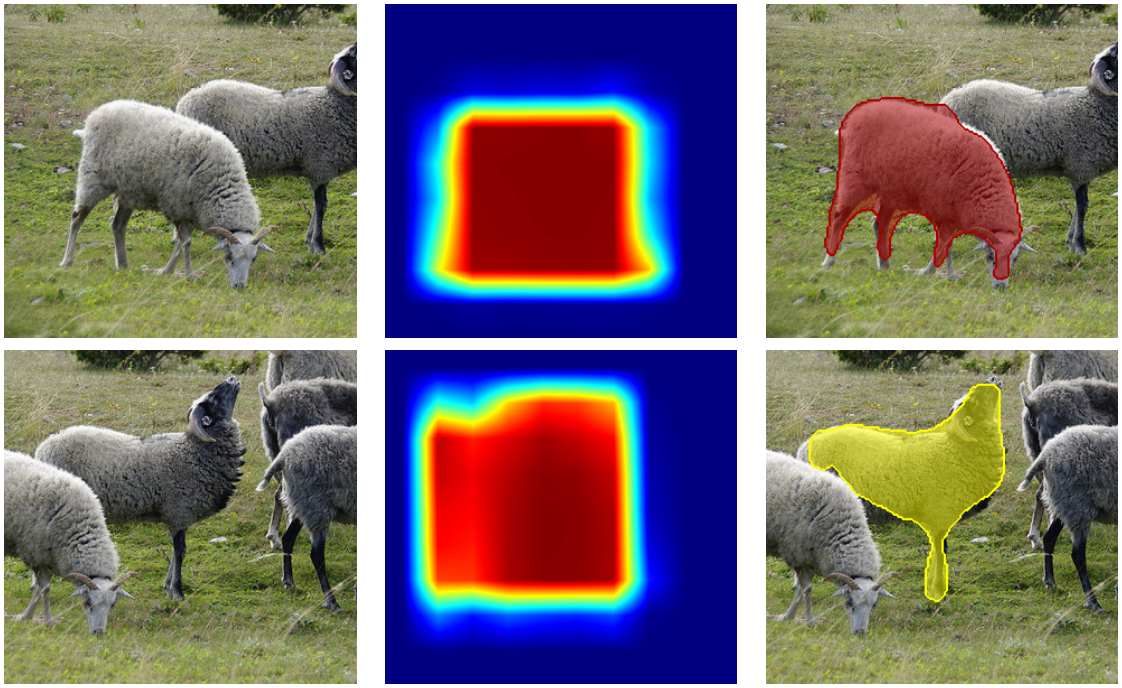
Figure 3.9: Overview of the one-shot system FastMask [Hu et al., 2017a]. The backbone network processes the entire input image only once (yellow). The resulting feature map is further downsampled using neck modules (orange), yielding a feature pyramid. Across the pyramid levels, all possible windows of a fixed size are extracted (green windows). Subsequently, a segmentation mask and an objectness score are generated for each window. Within the segmentation process, foreground attention (FG Attention) is utilized to prune background features. For clarity, we omit the pyramid levels $\mathcal{S}_{24}$, $\mathcal{S}_{48}$, $\mathcal{S}_{96}$, and $\mathcal{S}_{192}$. Input image taken from the COCO dataset [Lin et al., 2014].

objectness score to increase efficiency. Overall, this leads to an encoder-decoder architecture for the segmentation branch as visualized in Fig. 3.8(b), while the objectness scoring branch remains as in DeepMask.

Besides the refinement modules, Pinheiro et al. [2016] replace DeepMask's VGG-11 backbone with a ResNet-50 backbone without the final *conv5* stage (see Appendix A.2). Removing the *conv5* stage keeps the number of downsampling operations constant w.r.t. the previously used VGG-11 backbone. Due to the encoder-decoder structure, Pinheiro et al. [2016] also reduce the input window size to 160×160. Similar to DeepMaskZoom, Pinheiro et al. [2016] propose a variation of SharpMask denoted as SharpMaskZoom with 8 image pyramid levels.

FastMask Hu et al. [2017a] propose the one-shot object proposal generation system *FastMask*. The key methodological change compared to DeepMask and SharpMask is that the backbone network processes the entire image only once. Hence, FastMask utilizes a feature pyramid within the network following the one-shot concept. This transition from an image pyramid to a feature pyramid is visible in Fig. 3.7 and increases FastMask's computational efficiency while generating competitive results.

Similar to SharpMask, FastMask uses the same reduced ResNet-50 backbone. On top of the backbone, FastMask consists of a feature pyramid that uses the output of the backbone as the base level (see Fig. 3.9). This base level has a spatial resolution that is downsampled by a factor of 16 w.r.t. the input image. Thus, we call this pyramid level $\mathcal{S}_{16}$. From this level, additional pyramid levels, $\mathcal{S}_{32}$, $\mathcal{S}_{64}$, and $\mathcal{S}_{128}$ with respective downsampling factors, are created using residual neck modules (orange blocks in Fig. 3.9). The residual neck modules successively downsample the feature map utilizing average pooling, convolutional layers, and a skip connection. Each pyramid level contains a feature map for extracting objects of different sizes. From the feature maps at the different pyramid levels, FastMask extracts all possible 10×10 windows (green boxes in Fig. 3.9). Extracting fixed-size windows from feature maps of different resolutions (pyramid levels) leads to windows representing areas of different sizes in the input image. While 10×10 windows extracted from $\mathcal{S}_{16}$ represent small objects in the image, 10×10 windows extracted from $\mathcal{S}_{128}$ represent large objects. In addition to the pyramid levels $\mathcal{S}_{16}$, $\mathcal{S}_{32}$, $\mathcal{S}_{64}$, and $\mathcal{S}_{128}$, FastMask also includes the pyramid levels $\mathcal{S}_{24}$, $\mathcal{S}_{48}$, $\mathcal{S}_{96}$, and $\mathcal{S}_{192}$ to improve the results.



(a) Window in the image **(b)** Foreground attention **(c)** Segmentation mask

Figure 3.10: Examples showing the foreground attention in FastMask that separates the centered object from other objects and the background. The first column (a) shows the windows of the input image that are processed in FastMask's foreground attention module as feature representations. From the foreground attention in the second column (b), it is visible that the high-attention areas roughly capture the central sheep. Depending on the extracted window, FastMask attends to the sheep in the front (upper row) or the sheep in the back (lower row). The final column (c) shows the resulting segmentation masks, which only capture the respective central object. Input image windows taken from the COCO dataset [Lin et al., 2014].

All extracted windows compose a batch of possible proposals areas that are further processed in FastMask's head module. First, for every window an objectness score is generated using simple, fully connected layers similar to DeepMask and SharpMask (bottom right in Fig. 3.9). Second, for windows with a high objectness score, FastMask generates a segmentation mask in two steps. The first step roughly segments the centered object and prunes the background with a learned foreground attention. Figure 3.10 shows examples of the foreground attention indicating its strength to separate the central object from other objects and the background. Subsequently, FastMask generates a 40×40 segmentation mask based on the foreground attention and the 10×10 feature map of an extracted window. The foreground attention and the final segmentation mask are generated with per-pixel classifiers similar to DeepMask and SharpMask. Outside the network, the segmentation masks are upsampled and inserted into the image at their respective position.

To train FastMask, the three losses for the objectness ($\mathcal{L}_{\text{objn}}$), the foreground attention[4] ($\mathcal{L}_{\text{att}}$), and the segmentation mask ($\mathcal{L}_{\text{seg}}$) are utilized. All losses are cross-entropy losses. However,

[4]The ground truth for the foreground attention is the bounding box around the annotated object within the extracted window.

the losses for the foreground attention and the segmentation mask are normalized by the number of pixels in the window. Hence, the overall loss function is

$$\mathcal{L} = \mathcal{L}_{\text{objn}} + \mathcal{L}_{\text{att}} + \mathcal{L}_{\text{seg}}. \tag{3.12}$$

Similar to DeepMask and SharpMask, FastMask regards windows that wholly contain a centered object which fits the pyramid level as positive samples. For an object to fit a specific pyramid level, the object's height and width must be between 4 and 8 pixels on the respective pyramid level. Thus, for an object to fit $\mathcal{S}_{128}$, the object's height and width must be between $4 \cdot 128 = 512$ pixels and $8 \cdot 128 = 1024$ pixels in the input image. The rules for wholly contained objects and centered objects are similar to DeepMask and SharpMask. If one of the three conditions is not fulfilled, the window is considered a negative sample. This leads to a training strategy focusing on hard negative samples.

3.2.4 Discussion

We reviewed several object proposal generation approaches from three main classes. The window scoring-based approaches like Edge Boxes [Zitnick and Dollár, 2014] and BING [Cheng et al., 2014] efficiently generate box proposals by assessing a multitude of windows. However, those methods are unable to precisely capture the objects due to a missing refinement of the initial windows. Grouping-based approaches merge pixels or superpixels and produce detailed segmentation masks. To capture complex multi-colored objects, systems like Selective Search [Uijlings et al., 2013] use various features. This leads to a large pool of proposals and impedes the computational efficiency [Uijlings et al., 2013; Arbeláez et al., 2014]. In general, all box-based or mask-based approaches not utilizing CNNs have problems discovering small or medium objects. These problems arise due to the lack of small windows or too coarse superpixel segmentations. Additionally, most systems have problems generating proposal for entire complex objects, as already outlined in the introduction.

CNN-based approaches exist for both box-based and mask-based object proposal generation. Since the box-based approaches like RPN [Ren et al., 2016] are unable to produce segmentation masks by definition, we focus on the mask-based approaches. These systems are able to mitigate several typical problems of traditional systems, like capturing entire objects or a more efficient processing. However, the discovery of small objects remains challenging for CNN-based systems due to the downsampling process in CNNs. This downsampling process also leads to coarse proposals that do not adhere well to the object boundaries. Finally, apart from FastMask, most CNN-based approaches use an exhaustive sliding window approach, which is still computationally inefficient.

Since the CNN-based approaches produce superior results compared to the other classes, we focus on improving these systems in this thesis. Specifically, our proposed systems in Ch. 4 - Ch. 7 will follow the efficient one-shot concept of FastMask. Since discovering small objects is still challenging, we address this problem in Ch. 4. To bridge the gap between traditional superpixel grouping approaches and CNN-based methods, we will present a system that combines the advantages of both worlds in Ch. 5 - Ch. 7.

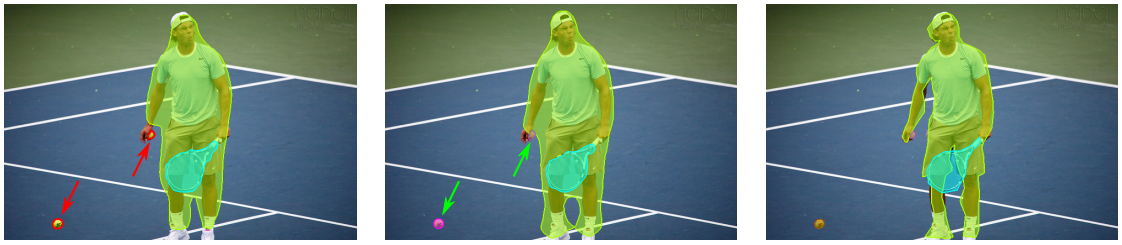
Chapter 4

AttentionMask: Attention-based Object Proposals

Table of Contents

As discussed in the introduction, the discovery of small objects is a main limitation of modern CNN-based object proposal generation systems [Pinheiro et al., 2015, 2016; Hu et al., 2017a]. This limitation is reasonable since even for humans, it is more challenging to discover small objects than large ones. For instance, the small tennis balls in Fig. 4.1 are harder to discover than the large tennis player. However, small objects like pens, paper-clips, or a computer mouse are ubiquitous. Similarly, more than 40% of the objects in the complex COCO dataset [Lin et al., 2014] are small due to their original size or the distance to the camera. As discussed previously, the discovery of small objects is challenging in CNN-based systems due to the inherent downsampling process in CNNs. This downsampling process is necessary to extract semantically rich features but also removes detailed spatial information from feature maps that are necessary to discover small objects.

Most CNN-based object proposal generation systems that produce masks extract windows from a pyramid to discover objects of different sizes (see. Sec. 3.2.3). Therefore, adding another level to the bottom of the pyramid for approaches like SharpMaskZoom and DeepMaskZoom [Pinheiro et al., 2016] would be a simple solution. However, the new pyramid level



(a) FastMask **(b)** AttentionMask **(c)** Ground truth

Figure 4.1: Results of FastMask [Hu et al., 2017a] (a) and our proposed AttentionMask (b) for an image with objects of different sizes. Comparing the results to the ground truth (c) shows that AttentionMask captures both small tennis balls (green arrows), while FastMask misses both of them (red arrows). This is the result of the new module for discovering small objects in AttentionMask. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) is visualized per annotated object. Input image and annotations taken from the COCO dataset [Lin et al., 2014].

would lead to a significant growth of possible windows and further increase the inefficiency of multi-shot approaches like DeepMask and SharpMask. Hence, we move to the efficient one-shot concept of FastMask [Hu et al., 2017a]. Adding a level to the bottom of the feature pyramid in FastMask poses three problems. First, the overall number of possible windows is almost four times larger, exceeding the memory available on modern GPUs[1]. Second, if enough GPU memory is available, most windows would only contain background. This is visible from the example in Fig. 4.2 that depicts a collage of every other window extracted from the base level of FastMask's feature pyramid. Although most windows cover background area, each window is processed in FastMask. Finally, since a ResNet-50 with four stages is utilized as the backbone, there is no natural location for adding another pyramid level in FastMask.

Humans face a similar problem in terms of efficiency since the retina generates around 250,000 times more data than we can consciously perceive[2] [Pierce and Karlin, 1957; Zhaoping, 2014]. To handle this imbalance, humans use visual attention to steer the capacity efficiently to relevant areas of the visual field and ignore the rest [Pashler, 1997]. For instance, in the model of Rensink [2000], only attention combines low-level proto-objects to form an object perception. This general capability of attention was also replicated by computational attention systems [Itti et al., 1998; Harel et al., 2006; Frintrop et al., 2015] and showed promising results in tasks like image thumbnailing [Marchesotti et al., 2009] or object tracking [Mahadevan and Vasconcelos, 2012].

Therefore, we exploit the concept of visual attention for object proposal generation. Specifically, we propose scale-specific objectness attention, visualized in Fig. 4.3, as an integral part of our novel, efficient object proposal generation system *AttentionMask*. AttentionMask follows the one-shot concept by extracting and processing windows from a feature pyramid to discover objects. The novel scale-specific objectness attention focuses the window extraction on promising areas within each level of the feature pyramid. Since large objects and small objects appear at different locations, an independent attention map is generated per pyramid level (scale-specific) as visible in Fig. 4.3. Additionally, the attention is designed to focus

[1] By modern GPUs, we refer to GPUs with 12 GB memory like the NVIDIA GeForce GTX TITAN X.

[2] Zhaoping [2014] argues that the retina emits $10^7 \frac{\text{bits}}{\text{s}}$ while Pierce and Karlin [1957] show in experiments that humans can consciously perceive around $40 \frac{\text{bits}}{\text{s}}$ during fast reading.

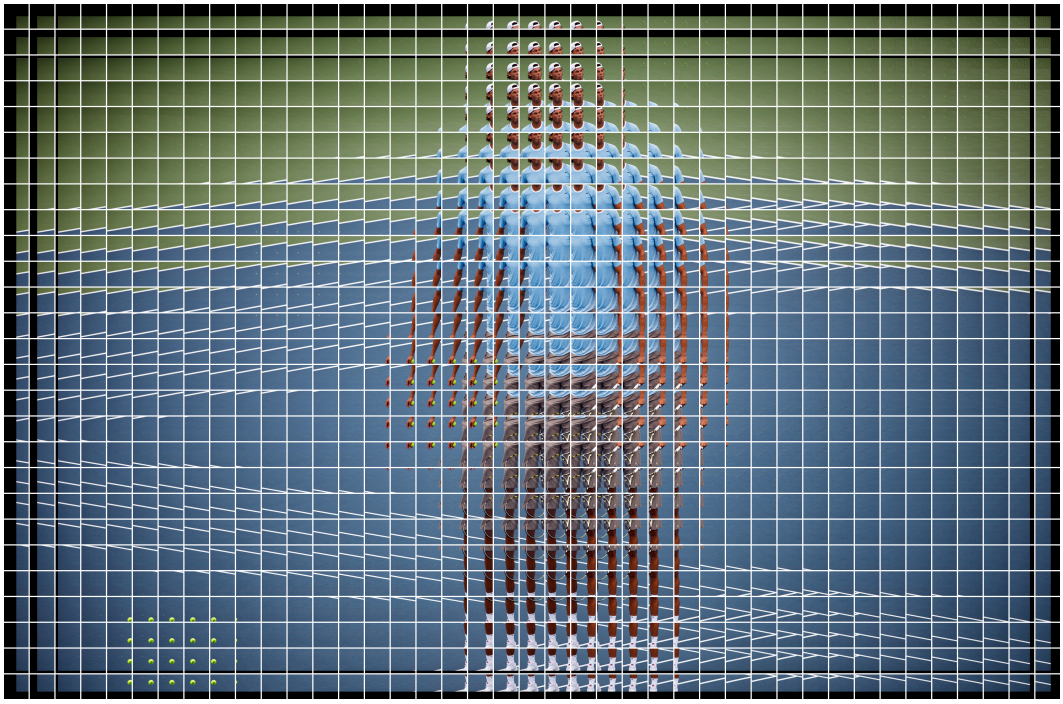


Figure 4.2: Visualization of the windows extracted from the base pyramid level of FastMask [Hu et al., 2017a] for discovering small objects. Most windows cover background areas and could easily be pruned early in the system to increase computational efficiency. For clarity, the visualization omits every other window in the horizontal and the vertical direction. Therefore, only a quarter of the actually extracted windows are shown. Base image taken from the COCO dataset [Lin et al., 2014].

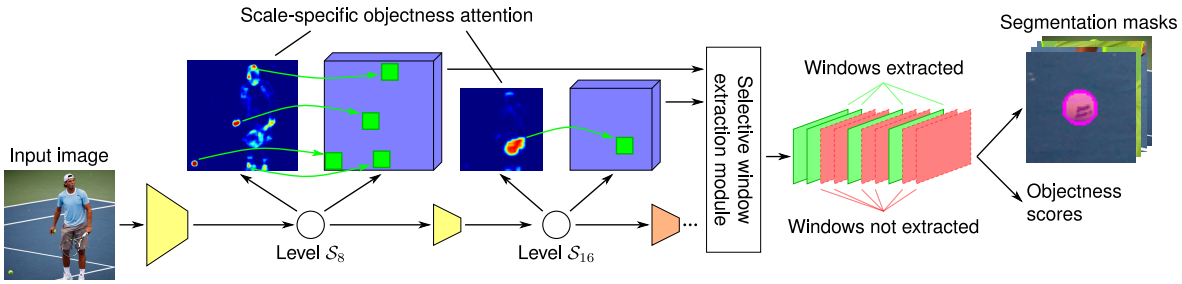


Figure 4.3: Overview of our object proposal generation system *AttentionMask*. First, a feature pyramid is constructed similar to FastMask, based on features from a backbone network (yellow). Scale-specific objectness attention is calculated at each pyramid level to highlight relevant regions for a selective window extraction. Using this efficient processing and the freed GPU resources, we add the module $\mathcal{S}_8$ to the feature pyramid as the new base level for an improved discovery of small objects. The final segmentation and objectness scoring are similar to FastMask and other systems. Input image taken from the COCO dataset [Lin et al., 2014].

on all objects of a specific pyramid level to comply with the class-agnostic processing in object proposal generation. Utilizing this scale-specific objectness attention in a one-shot approach allows computationally efficient processing with less memory consumption by pruning irrelevant windows early. Moreover, the pruning of windows also reduces false positives. In a second step, we exploit the more efficient processing pipeline and add a new module to the system for discovering small objects (pyramid level $\mathcal{S}_8$ in Fig. 4.3). This step is only possible due to the computationally efficient, attention-based processing.

This chapter presents our novel object proposal generation system AttentionMask based on our publication Wilms and Frintrop [2018]. We briefly discuss the use of attention in humans and computer vision in Sec. 4.1. Subsequently, we introduce the architecture of AttentionMask in Sec. 4.2 and discuss the training regime in Sec. 4.3. Highlighting the quality of AttentionMask to discover objects of all sizes, including small objects, we present a detailed evaluation on the complex COCO dataset [Lin et al., 2014] in Sec. 4.4. Finally, Sec. 4.5 discusses the main findings, contributions, and limitations of the system.

4.1 Attention in Humans and Computer Vision

As discussed above, humans use visual attention [Pashler, 1997] to direct their limited processing resources to relevant locations of the visual field. The process behind steering the attention is not entirely understood yet. However, psychological models exist that seek to represent the process and explain experimental findings [Treisman and Gelade, 1980; Wolfe et al., 1989; Rensink, 2000]. For instance, Treisman and Gelade [1980] argue that various features are extracted from the visual input. Subsequently, the saliency per location and feature is fused across different scales and features. The result is often referred to as a saliency map [Koch and Ullman, 1985] and highlights locations in the visual field that humans will likely attend to.

This general idea also serves as an inspiration for computational attention systems [Itti et al., 1998; Harel et al., 2006; Frintrop et al., 2015; Kruthiventi et al., 2017], which aim to mimic the effects of human visual attention. These models typically produce a saliency map that guides the processing of several computer vision systems addressing diverse tasks [Marchesotti et al., 2009; Mahadevan and Vasconcelos, 2012; Frintrop et al., 2014]. Besides the computational attention systems, recent CNN-based computer vision systems utilize attention in a purely data-driven and task-dependent framework [Xu et al., 2015; Yang et al., 2016; Chen et al., 2016a; Kong et al., 2017; Anderson et al., 2018]. Attention in these systems is learned end-to-end inside the CNN to reduce the computational load or prune distracting visual input. For instance, Yang et al. [2016] derive attention from an input question in visual question answering to prune visual features unrelated to the question in a CNN-based framework. Similarly, Chen et al. [2016a] learn attention to prune features of irrelevant scales for semantic segmentation, while Kong et al. [2017] select promising locations for object detection utilizing attention.

Overall, there is a long way from human visual attention to the data-drive and task-dependent attention utilized in CNNs. Our proposed scale-specific objectness attention is similar to the learned, data-driven attention used in CNNs, since we not only focus on salient regions but on all objects of a specific size. Hence, it is also different from object proposal generation approaches that locate salient regions and subsequently discover objects in these regions [Alexe et al., 2010; Frintrop et al., 2014; Martín García et al., 2015; Werner et al., 2015]. Most

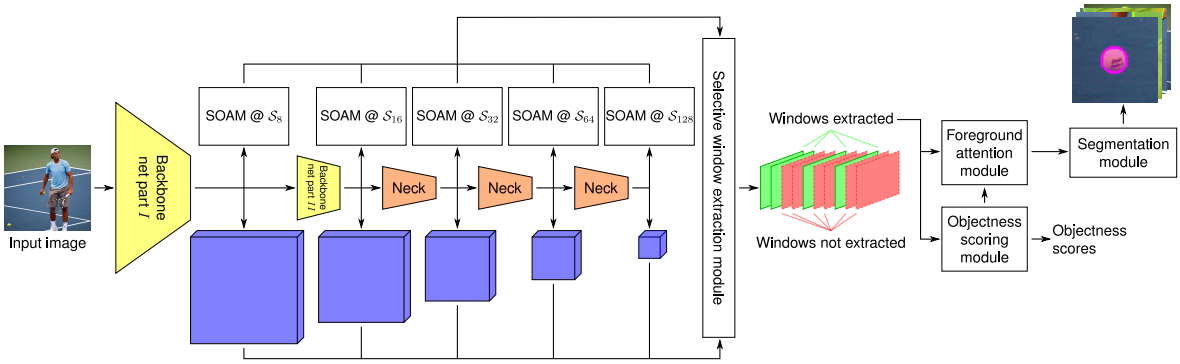


Figure 4.4: Detailed system figure of the proposed *AttentionMask* system. The backbone network (yellow) and the neck modules (orange) process the input image, yielding a feature pyramid. The backbone network is split into two parts, which allows the integration of the new pyramid level $\mathcal{S}_8$. At each pyramid level, one of our novel *Scale-specific Objectness Attention Modules* (SOAMs) extracts the scale-specific objectness attention for the selective window extraction. Thus, windows are only extracted from the feature maps (blue boxes) at high-attention locations. This leads to a batch of feature windows (green windows), omitting unnecessary background windows (red windows). For each extracted window, an objectness score is calculated, and a segmentation mask is generated utilizing foreground attention. Note that we omit the pyramid levels $\mathcal{S}_{24}$, $\mathcal{S}_{48}$, $\mathcal{S}_{96}$, and $\mathcal{S}_{192}$ for clarity. Input image taken from the COCO dataset [Lin et al., 2014].

similar to our scale-specific objectness attention are Chen et al. [2016a] and Kong et al. [2017] since they learn attention to focus on relevant scales and locations within a feature pyramid. However, we apply this principle to object proposal generation and utilize it to improve the discovery of small objects.

4.2 Attention-based Object Proposals

This section presents the architecture of our novel object proposal generation system AttentionMask (see Fig. 4.4). As briefly introduced above, AttentionMask follows the one-shot concept and therefore consists of a backbone network with a feature pyramid on top, described in Sec. 4.2.1. For each level of the pyramid, we efficiently calculate our new scale-specific objectness attention (SOAMs in Fig. 4.4) and extract fixed-size windows from high-attention areas of each pyramid level. We describe the scale-specific objectness attention generation and the window extraction in Sec. 4.2.2. To improve the discovery of small objects, we add a dedicated module ($\mathcal{S}_8$ in Fig. 4.4) as the new base level to the feature pyramid. The integration of this new module is detailed in Sec. 4.2.3, while the objectness scoring and segmentation are described in Sec. 4.2.4.

4.2.1 Backbone and Feature Pyramid

AttentionMask starts with a backbone network that extracts semantically rich features from the input image. Following Pinheiro et al. [2016] and Hu et al. [2017a], we use a ResNet-50 [He et al., 2016a] without the final *conv5* stage as the backbone (see Appendix A.2). The final stage is omitted to remove one downsampling operation from the backbone and preserve spatial details in the final feature map. Hence, the base of the feature pyramid, the backbone network's output, is a feature map that is downsampled by a factor 16 compared to the input images. We coin this pyramid level $\mathcal{S}_{16}$. Three more pyramid levels are derived from this base

level using residual neck modules [Hu et al., 2017a] (orange modules in Fig. 4.4). The neck modules downsample the feature map by a factor of 2 using average pooling with convolutional layers and a skip connection yielding pyramid levels $\mathcal{S}_{32}$, $\mathcal{S}_{64}$, and $\mathcal{S}_{128}$ as visualized in Fig. 4.4. Those feature maps are downsampled by a factor of 32, 64, and 128 compared to the input image.

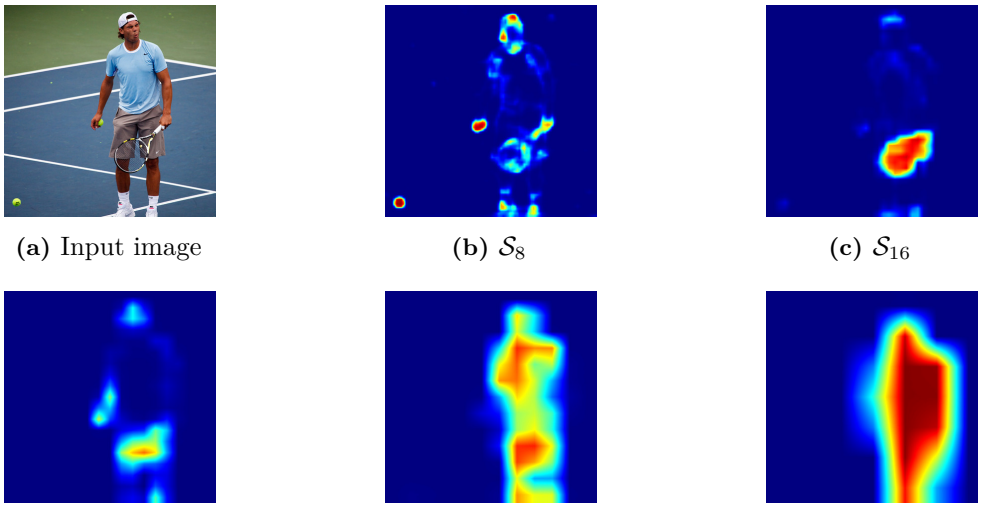
To improve the discovery of objects, we use four optional, intermediate pyramid levels similar to FastMask [Hu et al., 2017a]. The additional levels represent the image downsampled by a factor of 24, 48, 96, and 192. Hence, those levels are integrated between the previously described pyramid levels or after the final pyramid level. Technically, the intermediate pyramid levels are produced by duplicating the *conv4* stage of the backbone network and replacing the stride 2 convolution with a stride 3 convolution. This change in stride leads to the pyramid level $\mathcal{S}_{24}$. Subsequently, $\mathcal{S}_{48}$, $\mathcal{S}_{96}$, and $\mathcal{S}_{192}$ are derived from $\mathcal{S}_{24}$ using multiple neck modules as described above.

Overall, the backbone network leads to a feature pyramid with up to 8 levels. The pyramid levels are denoted as $\mathcal{S}_n$ with n expressing the downsampling factor w.r.t. the input image. At each pyramid level, a feature map with semantically rich features exists.

4.2.2 Scale-specific Objectness Attention

To discover objects of different sizes in the one-step concept, we extract fixed-size windows from locations across all levels of the feature pyramid. However, unlike FastMask, we do not extract all possible windows, since it is computationally expensive and leads to the extraction of many background windows as described above. Therefore, we integrate a *Scale-specific Objectness Attention Module* (SOAM) at each pyramid level of AttentionMask as visualized in Fig. 4.4. The SOAMs calculate the scale-specific objectness attention and highlight locations within each pyramid level that likely feature objects of sizes matching the pyramid level. Figure 4.5 shows example outputs of the SOAMs at the pyramid levels $\mathcal{S}_8$, $\mathcal{S}_{16}$, $\mathcal{S}_{32}$, $\mathcal{S}_{64}$, and $\mathcal{S}_{128}$. Red colors indicate high attention areas, while blue colors denote low attention areas. Based on these results, we extract 10×10 windows of the feature maps at high-attention locations with a new selective window extraction module. This selective extraction omits large background areas and reduces the computational load for the subsequent objectness scoring and segmentation steps.

To generate the scale-specific objectness attention, the SOAMs use a lightweight structure with only two convolutional layers (see Fig. 4.6). The first convolutional layer applies 128 kernels of size 3×3 with ReLU, while the second convolutional layer applies two 4×4 kernels with a subsequent softmax. Generally, we tackle the problem in a semantic segmentation framework with the classes *object* and *non-object* for each location between four pixels of a pyramid level. The locations are between pixels since we extract windows with even side lengths (10×10) following FastMask. Accordingly, the architecture of the SOAMs uses unusual 4×4 kernels in the second convolutional layer. The presented SOAM architecture provides the best results on the overall object proposal generation task. This is visible from the results in Tab. 4.1, which compares different SOAM architectures. The lower part of Tab. 4.1 also shows that using only one SOAM across all pyramid levels degrades the results. The one SOAM would generate an attention map covering objects of all sizes. As a result, large objects covering most of the image lead to unnecessarily extracted windows on the lower levels of the pyramid. This increases the runtime and induces additional false positives.



(a) Input image (b) $\mathcal{S}_8$ (c) $\mathcal{S}_{16}$

(d) $\mathcal{S}_{32}$ (e) $\mathcal{S}_{64}$ (f) $\mathcal{S}_{128}$

Figure 4.5: Scale-specific objectness attention maps for the pyramid levels $\mathcal{S}_8$ (see Sec. 4.2.3), $\mathcal{S}_{16}, \mathcal{S}_{32}, \mathcal{S}_{64}$, and $\mathcal{S}_{128}$ (b) - (f), given the input image (a). The scale-specific objectness attention maps at different pyramid levels highlight objects or object parts of different sizes. For instance, the map at pyramid level $\mathcal{S}_{16}$ highlights the racket, while the map at pyramid level $\mathcal{S}_{128}$ highlights the player. All scale-specific objectness attention maps are upsampled to input image size for improved visibility. Red areas indicate high attention, while blue areas indicate low attention. Input image taken from the COCO dataset [Lin et al., 2014].

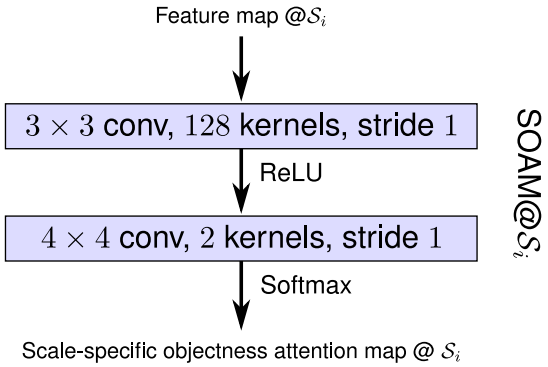
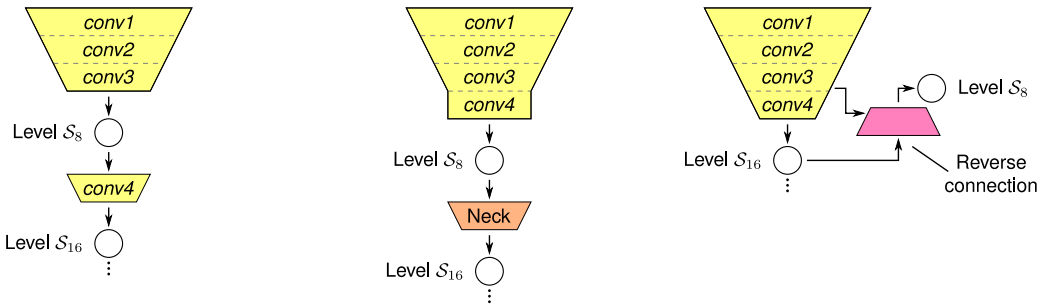


Figure 4.6: Architecture of the lightweight *Scale-specific Objectness Attention Module* (SOAM) that extracts the scale-specific objectness attention at every pyramid level $\mathcal{S}_i$ of AttentionMask's feature pyramid to guide the subsequent window extraction.

Table 4.1: Variations of the SOAM architecture. The first column describes the architecture variations with the number of kernels per layer and the kernel sizes in parentheses. AR@100 denotes the Average Recall for the first 100 proposals on our COCO validation dataset. We utilize AttentionMask$_{\text{val}}$ with the respective SOAM architecture without $\mathcal{S}_8$. The GPU runtime only includes the processing of the network's forward pass. The chosen architecture is highlighted in **bold font**.

Architecture	AR@100↑	GPU runtime↓
$2(4 \times 4)$	0.258	$0.20s$
$\mathbf{128(3 \times 3) - 2(4 \times 4)}$	**0.261**	**0.21s**
$128(3 \times 3) - 128(3 \times 3) - 2(4 \times 4)$	0.260	$0.23s$
$256(3 \times 3) - 2(4 \times 4)$	0.261	$0.23s$
$2(4 \times 4)$, one SOAM for all pyramid levels	0.216	$0.25s$



(a) Splitting the backbone network

(b) *conv4* stage with dilated convolution

(c) Reverse connection

Figure 4.7: Three architectures for adding the new pyramid level $\mathcal{S}_8$ to AttentionMask. (a): The backbone network is split to generate $\mathcal{S}_8$ from *conv3* features. (b): Dilated convolution prevents further downsampling in the *conv4* stage and leads to $\mathcal{S}_8$ at the end of the backbone network. (c): A reverse connection [Kong et al., 2017] upsamples the *conv4* features and integrates the earlier *conv3* features with higher spatial resolution.

Note that the foreground attention proposed in FastMask and also utilized in AttentionMask solves a different problem. As described in Sec. 3.2.3, the foreground attention limits the influence of features within an extracted window that do not belong to the central object. Hence, the general idea of pruning the background is similar. However, the foreground attention is utilized at a different system stage. Moreover, foreground attention is a soft attention that weights features, while scale-specific objectness attention is a hard attention that excludes windows.

In a nutshell, the SOAMs focus the window extraction on promising areas of each pyramid level. As a result, fewer windows are extracted, resulting in a computationally efficient processing pipeline and fewer false positives. Each of the extracted windows will subsequently serve as the base for generating one object proposal.

4.2.3 Discovery of Small Objects

The previously introduced SOAMs lead to an efficient processing pipeline with reduced runtime and memory consumption on the GPU by omitting the extraction of background windows. We utilize this efficient processing and the freed GPU resources to discover objects that are too small for the pyramid level $\mathcal{S}_{16}$[3], like the two tennis balls in Fig. 4.1. To this end, we propose a new base level $\mathcal{S}_8$ with downsampling factor 8 for AttentionMask's feature pyramid. Thus, the resolution of the base level in the feature pyramid increases by a factor of 2. If we consider an input image of size 800×600, the addition of $\mathcal{S}_8$ without the previously proposed SOAMs would lead to 7500 additionally extracted windows. This is almost 3 times more than the number of extracted windows from all other pyramid levels together. Hence, adding $\mathcal{S}_8$ to the system without the SOAMs is impossible due to memory limitations on modern GPUs as discussed above.

To add $\mathcal{S}_8$ to the feature pyramid, we split the backbone between the stages *conv3* and *conv4* as visualized in Fig. 4.4 and in more detail in Fig. 4.7(a). Since the result of stage *conv3* is only

[3]Objects are too small if they do not fit the size definition of $\mathcal{S}_{16}$ during training. The lower limit for an object's horizontal and vertical dimensions to fit $\mathcal{S}_{16}$ is 64 pixels at input image size.

Table 4.2: Architectures for adding $\mathcal{S}_8$ to AttentionMask based on splitting the backbone network, dilated convolution, and a reverse connection. The first column describes the architecture variations. AR@100 denotes the Average Recall for the first 100 proposals on our COCO validation dataset. We utilize AttentionMask$_{\text{val}}$ with the respective $\mathcal{S}_8$ integration (see Fig. 4.7 for visualizations). The GPU runtime only includes the processing of the network's forward pass. The chosen architecture is highlighted in **bold font**.

Architecture	AR@100↑	GPU runtime↓
Splitting the backbone network	**0.287**	**0.21s**
Dilated convolution	0.292	0.30s
Reverse connection	0.285	0.26s

downsampled by a factor of 8 w.r.t. the input image, we can use these features directly for $\mathcal{S}_8$. However, using features from earlier stages of the backbone might degrade the results, since they are not as semantically rich as the features from the final stage (*conv4*). Therefore, we compare our proposed approach with two other strategies. First, we replace the strided convolution at the beginning of backbone's *conv4* stage with a dilated convolution (dilation factor 2). This dilated convolution extends the receptive field without reducing the spatial resolution and is frequently applied in semantic segmentation [Chen et al., 2015a; Yu and Koltun, 2016; Chen et al., 2017]. Hence, the result of the backbone network is only downsampled by a factor of 8 (see Fig. 4.7(b)). Second, we upsample the features of the *conv4* stage using a reverse connection proposed by Kong et al. [2017] to create $\mathcal{S}_8$ from $\mathcal{S}_{16}$. The reverse connection combines deconvolved features from the *conv4* stage and features from the *conv3* stage as visualized in Fig. 4.7(c). The results in Tab. 4.2 comparing the three approaches indicate that our proposed approach exhibits the best trade-off between computational efficiency and effectiveness. Note that the dilated convolution approach is not applicable with 8 or 9 pyramid levels due to memory limitations on modern GPUs[4].

Overall, adding the new pyramid level $\mathcal{S}_8$ allows AttentionMask to discover smaller objects that are missed by systems like FastMask due to methodological limitations. To combine efficiency and effectiveness, we split the backbone network to integrate $\mathcal{S}_8$ into the feature pyramid. Note that the addition of $\mathcal{S}_8$ without reducing the number of extracted windows utilizing our novel SOAMs would be impossible due to limited GPU resources.

4.2.4 Objectness and Segmentation

The previously described steps extract 10×10 windows across all levels of AttentionMask's feature pyramid guided by the SOAMs. The extracted windows constitute a batch for further processing (green windows in Fig. 4.4). For each window, the objectness scoring module generates an objectness score utilizing three fully connected layers in a binary classification framework as in FastMask. Subsequently, the windows with the n highest objectness scores are segmented, leading to the final object proposals. We follow FastMask for creating the segmentation masks and first generate the foreground attention to focus on the central object. Guided by the foreground attention, we use two fully connected layers to generate segmentation masks of size 40×40 independent of the objects' sizes. These masks are further upsampled to

[4]We tested this configuration on a 12 GB GPU with the framework *Caffe*.

Table 4.3: Versions of AttentionMask with different pyramid levels $\mathcal{S}_8$ - $\mathcal{S}_{196}$. ✓ denotes that the pyramid level is included, while ✗ marks the omission of the level.

Version	Number of pyramid levels	Pyramid levels								
		$\mathcal{S}_8$	$\mathcal{S}_{16}$	$\mathcal{S}_{24}$	$\mathcal{S}_{32}$	$\mathcal{S}_{48}$	$\mathcal{S}_{64}$	$\mathcal{S}_{96}$	$\mathcal{S}_{128}$	$\mathcal{S}_{192}$
$\text{AttentionMask}_{\text{val}}$	5	✓	✓	✗	✓	✗	✓	✗	✓	✗
$\text{AttentionMask}_8^{128}$	8	✓	✓	✓	✓	✓	✓	✓	✓	✗
$\text{AttentionMask}_8^{192}$	9	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\text{AttentionMask}_{16}^{192}$	8	✗	✓	✓	✓	✓	✓	✓	✓	✓

size 160×160. The foreground attention task and the segmentation task are formulated as binary classifications of each pixel inside the windows.

Overall, the output of AttentionMask is a batch of up to n object proposals with pixel-precise segmentation masks and objectness scores. For our experiments, we set $n = 1000$.

4.2.5 AttentionMask Versions

In this chapter, we use four different versions of AttentionMask (see Tab. 4.3). The versions differ in the pyramid levels utilized for window extraction. The model $\text{AttentionMask}_{\text{val}}$ only uses the five main pyramid levels $\mathcal{S}_8$, $\mathcal{S}_{16}$, $\mathcal{S}_{32}$, $\mathcal{S}_{64}$, and $\mathcal{S}_{128}$. This model is exclusively utilized for ablation experiments and justifying design choices. $\text{AttentionMask}_8^{128}$ additionally includes $\mathcal{S}_{24}$, $\mathcal{S}_{48}$, and $\mathcal{S}_{96}$, yielding a feature pyramid with eight levels. This is the standard AttentionMask version and will be used in the subsequent chapters as AttentionMask unless otherwise noted. Adding the pyramid level $\mathcal{S}_{192}$ to the feature pyramid, like in FastMask, leads to $\text{AttentionMask}_8^{192}$, a model with nine pyramid levels. This is one level more than DeepMaskZoom, SharpMaskZoom and FastMask utilize. Finally, we propose $\text{AttentionMask}_{16}^{192}$, which corresponds to FastMask but uses our novel SOAMs. All three versions with eight or nine pyramid levels are evaluated in Sec. 4.4.

4.3 Training

After describing the architecture of AttentionMask, we discuss the end-to-end training regime. First, we describe the ground truth selection for the four different types of ground truth necessary to train AttentionMask in Sec. 4.3.1. Subsequently, Sec. 4.3.2 presents the individual and overall loss functions. Finally, we present the hyperparameters and the overall training process in Sec. 4.3.3.

4.3.1 Ground Truth Selection

AttentionMask needs ground truth for the objectness scoring module, the foreground attention module, the segmentation module, and the SOAMs. For the ground truth utilized in the first three modules, we follow FastMask [Hu et al., 2017a]. Hence, an extracted window is regarded as a positive sample for the objectness score and relevant for the foreground

Table 4.4: Comparison of different strategies to handle the imbalance between positive and negative samples in the ground truth for training the SOAMs. AR@100 denotes the Average Recall for the first 100 proposals on our COCO validation dataset. We utilize $\text{AttentionMask}_{\text{val}}$ with the respective strategy. The chosen strategy is highlighted in **bold font**.

Strategy	AR@100↑
No handling of class imbalance	0.186
Class weighting with the inverse class probability	0.279
Negative sample mining $1:3$ [Kong et al., 2017]	**0.287**

attention and segmentation tasks if it wholly contains a centered, annotated object. Centered means that the object's center is on one of the four central pixels of the 10×10 window. Additionally, the annotated object's size must fit the pyramid level of the window. To fit the pyramid level, we demand that the annotated object's height and width are 4 to 8 pixels at the respective pyramid level. All windows that satisfy these conditions constitute a pool of positive samples for the objectness score. From the pool of positive samples, up to p samples are randomly chosen and utilized in the training process. Similarly, p negative samples are chosen to train the objectness scoring module. A sample is negative if exactly one of the three conditions above does not apply [Hu et al., 2017a]. We follow FastMask and set $p = 32$ in our experiments.

To train the objectness scoring module, only the label *object* (positive sample) or *no object* (negative sample) is necessary per sample. In contrast, the annotated object mask for each positive sample is necessary to train the foreground attention module and the segmentation module. The foreground attention is trained with the bounding box around the annotated object mask as a rough segmentation. Only the actual segmentation module is trained with the precise annotated object mask. All pixels in the 10×10 (foreground attention) or 160×160 (segmentation) window that belong to the object (box or mask) are positive samples, while the remaining pixels are negative samples. Overall, training these three modules is similar to FastMask.

For training the SOAMs, we select the ground truth based on a similar selection process as outlined above. However, only the size of the annotated object has to fit the pyramid level. The other two conditions are irrelevant, since no windows are used. Hence, all locations of a pyramid level that belong to at least one annotated object that fits the pyramid level are positive samples. The remaining pixels constitute a pool of negative samples. Due to the high imbalance between the positive and negative samples ($1:138$ across all pyramid levels for the COCO training set), we use negative sample mining similar to Kong et al. [2017]. This strategy reduces the imbalance to $1:3$ or less by randomly selecting negative samples and is more effective compared to class weighting or ignoring the imbalance (see Tab. 4.4).

4.3.2 Loss Function

AttentionMask utilizes ground truth for four different modules in the network: the objectness scoring module, the foreground attention module, the segmentation module, and the proposed SOAMs. To select loss functions for the four modules, we follow FastMask and use binary

cross-entropy loss as well as normalized binary cross-entropy loss. The binary cross-entropy without normalization is defined as

$$L(p, g) = p \cdot -\log(g) + (1 - p) \cdot -\log(1 - g), \tag{4.1}$$

with the prediction p and the ground truth g. Since generating the objectness score is framed as a binary classification task per window, we use the binary cross-entropy loss following Hu et al. [2017a] and define the loss for the objectness scoring module $\mathcal{L}_{objn}$ as

$$\mathcal{L}_{\text{objn}}(p_{\text{objn}}, g_{\text{objn}}) = L(p_{\text{objn}}, g_{\text{objn}}). \tag{4.2}$$

While g_{objn} represents the label for the windows (0/1), p_{objn} is the predicted objectness.

Similarly, the losses for the foreground attention module $\mathcal{L}_{\text{att}}$ and the segmentation module $\mathcal{L}_{\text{seg}}$ are defined using binary cross-entropy loss. In both cases, the binary cross-entropy loss is calculated for each location in the output. We follow Hu et al. [2017a] and normalize the summed loss across the window by the window's size to scale the loss. This normalization leads to the normalized binary cross-entropy loss. Hence, the foreground attention loss $\mathcal{L}_{\text{att}}$ for a foreground attention ground truth $g_{\text{att}}(x, y)$ and the predicted foreground attention $p_{\text{att}}(x, y)$ is defined as

$$\mathcal{L}_{\text{att}}(p_{\text{att}}, g_{\text{att}}) = \frac{1}{10^2} \sum_{x=1}^{10} \sum_{y=1}^{10} L(p_{\text{att}}(x, y), g_{\text{att}}(x, y)). \tag{4.3}$$

Note that the foreground attention is always computed on 10×10 windows. The loss for the segmentation masks $\mathcal{L}_{\text{seg}}$, which is calculated based on 160×160 windows, is defined accordingly for an annotated object's segmentation mask $g_{\text{seg}}(x, y)$ and the predicted segmentation mask $p_{\text{seg}}(x, y)$ as

$$\mathcal{L}_{\text{seg}}(p_{\text{seg}}, g_{\text{seg}}) = \frac{1}{160^2} \sum_{x=1}^{160} \sum_{y=1}^{160} L(p_{\text{seg}}(x, y), g_{\text{seg}}(x, y)). \tag{4.4}$$

Since the SOAMs also solve a per-pixel binary classification task, we utilize normalized binary cross-entropy loss for the SOAMs. Unlike for the previous losses, the size of the output attention map h_i and w_i is not fixed and differs between SOAMs and images. Moreover, to incorporate the negative sample mining, which balances positive and negative samples, we set the loss of the ignored locations to 0 using the indicator function $\mathbf{1}_{\text{samples}}(x, y)$. Overall, the loss $\mathcal{L}_{SOAM_i}$ for the SOAM at pyramid level $\mathcal{S}_i$ is defined as

$$\mathcal{L}_{\text{SOAM}_i}(p_{\text{SOAM}_i}, g_{\text{SOAM}_i}) = \frac{1}{h_i \cdot w_i} \sum_{x=1}^{h_i} \sum_{y=1}^{w_i} \mathbf{1}_{\text{samples}}(x, y) L(p_{\text{SOAM}_i}(x, y), g_{\text{SOAM}_i}(x, y)). \tag{4.5}$$

The ground truth and the prediction for the SOAM at pyramid level $\mathcal{S}_i$ are denoted as g_{SOAM_i} and p_{SOAM_i}.

Combining the losses $\mathcal{L}_{\text{objn}}$, $\mathcal{L}_{\text{att}}$, $\mathcal{L}_{\text{seg}}$, and the losses $\mathcal{L}_{\text{SOAM}_i}$ for the different pyramid levels, we use a weighted sum for optimized results. Hence, the overall loss for AttentionMask is

$$\mathcal{L} = w_{\text{objn}}\mathcal{L}_{\text{objn}} + w_{\text{att}}\mathcal{L}_{\text{att}} + w_{\text{seg}}\mathcal{L}_{\text{seg}} + w_{\text{SOAM}} \sum_i \mathcal{L}_{\text{SOAM}_i}. \tag{4.6}$$

The weights w_{objn}, w_{att}, w_{seg}, and w_{SOAM} balance the influence of the individual losses. In the experiments, we use the following values: $w_{\text{objn}} = 0.5, w_{\text{att}} = 1.25, w_{\text{seg}} = 1.25$, and $w_{\text{SOAM}_i} = 0.25$.

4.3.3 Hyperparameters and Solver

To train AttentionMask based on the ground truth and losses introduced above, we use stochastic gradient descent with a learning rate of 0.0001, a momentum of 0.9, a weight decay of 0.00005, and batch size 1. We set the batch size to 1 following FastMask since we create a batch of extracted windows within the network. Hence, the batch size is 1 at the beginning of the network. However, for the later modules, it is higher and equals the number of extracted windows for one training image (up to 64). To improve the final results, we reduce the learning rate by a factor of $\frac{1}{10}$ once the validation loss does not improve for three consecutive epochs.

Overall, we train AttentionMask for 17 epochs on the COCO training dataset. The backbone network is initialized with ImageNet weights [He et al., 2016a], while the rest of the network is learned from scratch. To compensate for the different initial weights, we multiply the learning rate for the layers learned from scratch with a factor of 10.

4.4 Experiments

This section evaluates our proposed AttentionMask system and compares it to nine state-of-the-art object proposal generation approaches. The evaluation protocol follows previous work on object proposal generation [Pinheiro et al., 2015, 2016; Hu et al., 2017a]. All systems train on the training set of the COCO dataset [Lin et al., 2014], while we generate the reported results on the challenging COCO test dataset (see Sec. 2.2.2). To compare the different systems, we use Average Recall (AR, see Sec. 2.2.3) for different numbers of proposals to assess how many objects are discovered and how precisely they are captured. Moreover, we utilize variations of AR focusing on different object sizes for a more detailed analysis (see Tab. 2.1).

We compare the three AttentionMask versions AttentionMask$_8^{128}$, AttentionMask$_8^{192}$, and AttentionMask$_{16}^{192}$ to seven object proposal generation methods that create segmentation masks and eight methods generating boxes. For mask proposals, we compare to the CNN-based methods FastMask [Hu et al., 2017a], which has many architectural similarities with AttentionMask, SharpMaskZoom [Pinheiro et al., 2016], SharpMask [Pinheiro et al., 2016], DeepMaskZoom [Pinheiro et al., 2015, 2016], and InstanceFCN [Dai et al., 2016]. Moreover, we compare to MCG [Arbeláez et al., 2014; Pont-Tuset et al., 2017] and COB [Maninis et al., 2016, 2017], which do not use CNNs for proposal generation. All methods have recently shown strong results on the COCO dataset. The evaluation based on box proposals compares the AttentionMask versions to the same systems except for InstanceFCN[5]. Instead, we additionally compare to the methods BING [Cheng et al., 2014] and Edge Boxes [Zitnick and Dollár, 2014], which do not use CNNs but generate object proposals efficiently. See Sec. 3.2 for an in-depth

[5]Neither official code nor results on box proposals are publicly available.

Table 4.5: Results on the COCO test dataset using pixel-precise segmentation mask proposals in terms of six Average Recall (AR) measures and the runtime. AR^S, AR^M, and AR^L denote results on small, medium, and large objects. See Tab. 2.1 for details on the AR variations. **Bold font** highlights the best results, while *italic font* indicates the second-best results. †: Not including generation of edge maps. *: GPU-only runtime. The runtime for InstanceFCN is taken from Hu et al. [2017a].

Method	AR@10↑	AR@100↑	AR@1k↑	AR^S@100↑	AR^M@100↑	AR^L@100↑	Runtime↓
MCG	0.077	0.186	0.299	0.041	0.182	0.435	45s
COB	0.104	0.233	0.383	0.065	0.243	0.501	45s†
DeepMaskZoom	0.151	0.286	0.371	0.093	0.389	0.466	1.35s*
SharpMask	0.154	0.278	0.360	0.035	0.399	0.513	1.03s*
SharpMaskZoom	0.156	0.304	0.401	0.099	0.412	0.495	2.02s*
InstanceFCN	0.166	0.317	0.392	-	-	-	1.50s*
FastMask	0.169	0.313	0.406	0.106	0.406	0.517	0.33s*
$\text{AttentionMask}_8^{128}$	*0.180*	*0.349*	*0.444*	**0.162**	0.421	0.560	*0.22*s*
$\text{AttentionMask}_8^{192}$	**0.183**	**0.355**	**0.450**	*0.157*	*0.426*	*0.590*	*0.22*s*
$\text{AttentionMask}_{16}^{192}$	0.176	0.336	0.412	0.097	**0.438**	**0.594**	**0.21**s*

discussion of the different methods. All results except for InstanceFCN were generated in a controlled environment with an NVIDIA GeForce GTX TITAN X.

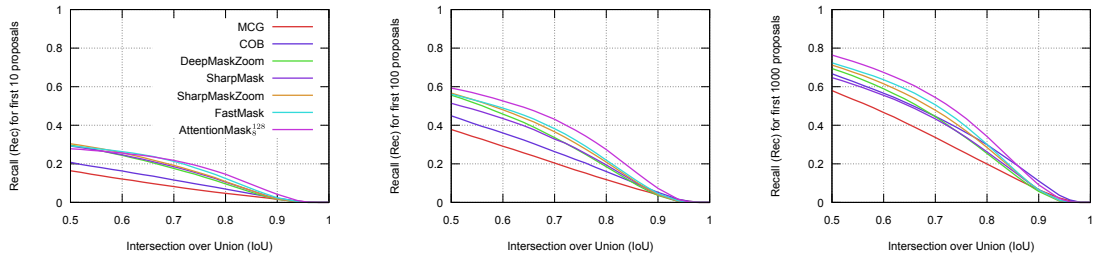
In the following sections, we first discuss the quantitative and qualitative results for the pixel-precise mask proposals in Sec. 4.4.1 and the quantitative results for the box proposals in Sec. 4.4.2. Subsequently, we give a detailed analysis of AttentionMask's GPU runtime and compare it to the other CNN-based systems in Sec. 4.4.3. Finally, Sec. 4.4.4 discusses the influence of the new pyramid level $\mathcal{S}_8$ and the SOAMs in more detail. Note that we already evaluated design choices like the architecture of the SOAMs in Sec. 4.2 and Sec. 4.3.

4.4.1 Results with Pixel-precise Mask Proposals

Quantitative Results

Table 4.5 presents the quantitative results of the different systems generating pixel-precise segmentation mask proposals on the challenging COCO test dataset. The results show that the AttentionMask versions outperform all other systems, including FastMask, in terms of all six AR measures and the runtime. $\text{AttentionMask}_8^{128}$, the standard version of AttentionMask, improves by 11.5% over FastMask and 14.8% over SharpMaskZoom in terms of AR@100. This improvement is driven by the enhanced performance on small objects (AR^S@100) based on the new pyramid level $\mathcal{S}_8$. In terms of AR^S@100 the improvements are 52.8% (FastMask) and 63.6% (SharpMaskZoom). However, $\text{AttentionMask}_8^{128}$ outperforms the other systems also on medium and large objects by utilizing our SOAMs. Hence, the introduction of the SOAMs already improves the performance. This is supported by the improvement of $\text{AttentionMask}_{16}^{192}$ over FastMask (7.3% in terms of AR@100) that use identical pyramid levels.

Comparing AttentionMask to SharpMask, DeepMaskZoom, InstanceFCN, MCG, and COB shows even more significant improvements. Compared to SharpMask, which does not utilize an extra pyramid level for smaller objects, the improvement on small objects (AR^S@100)



(a) Recall@10 **(b)** Recall@100 **(c)** Recall@1000

Figure 4.8: Recall (Rec) of selected object proposal generation systems for the first 10 proposals (a), 100 proposals (b), and 1000 proposals (c) across different IoU values on the COCO test dataset.

is 363% (AttentionMask$_8^{128}$). MCG and COB, the only methods not utilizing CNNs during object proposal generation in this evaluation, exhibit considerably lower numbers across most AR values. The strong results of AttentionMask compared to all other methods are also visible from the plots in Fig. 4.8, which show the Recall (Rec) for 10, 100, and 1000 proposals across different IoU levels. Similar to the results in Tab. 4.5, AttentionMask$_8^{128}$ outperforms all other methods across almost all IoU levels. Overall, AttentionMask discovers 76% of the objects with the first 1000 object proposals (Rec(0.5) in Fig. 4.8(c)).

Focusing on the three versions of AttentionMask, the results in Tab. 4.5 show improved results on small objects if the new pyramid level $\mathcal{S}_8$ is utilized. AttentionMask$_8^{192}$ outperforms AttentionMask$_{16}^{192}$ by 61.9% in terms of AR^S@100, highlighting the strong influence of $\mathcal{S}_8$ for discovering small objects. Adding $\mathcal{S}_{192}$ to the feature pyramid leads to improved results on large objects (+5.3%, AttentionMask$_8^{128}$ vs. AttentionMask$_8^{192}$). However, the results on small objects decrease by 3.1% since the new proposals replace existing ones. Similar effects are visible for medium and large objects when comparing AttentionMask$_8^{192}$ with AttentionMask$_{16}^{192}$. Overall, utilizing a fair setup with eight pyramid levels similar to other systems (FastMask, SharpMaskZoom, DeepMaskZoom), AttentionMask$_8^{128}$ exhibits the best results.

Qualitative Results

Figure 4.9 shows qualitative results of AttentionMask$_8^{128}$, FastMask, and SharpMaskZoom on images from the COCO test set to complement the quantitative results. The results indicate that AttentionMask discovers more small objects than the other systems, which is in line with the previous findings. For instance, the sheep in the first row and the birds in the second row are hard to recognize for humans as well. Nevertheless, AttentionMask discovers all the objects, while FastMask and SharpMaskZoom miss most small sheep and birds. Similarly, the two tennis balls in the third row are not discovered by FastMask and SharpMaskZoom, despite the simple background. AttentionMask captures both tennis balls utilizing $\mathcal{S}_8$.

Rows four and five show examples with more complex scene compositions, including a high level of clutter. In the cluttered classroom environment (fourth row), AttentionMask discovers almost all students, chairs, and laptops, while FastMask and SharpMaskZoom miss most small chairs and some smaller laptops. In the fifth row, AttentionMask discovers almost all small glasses on the shelf in a cluttered kitchen environment. In contrast, FastMask and SharpMaskZoom discover at most one instance. However, AttentionMask misses one

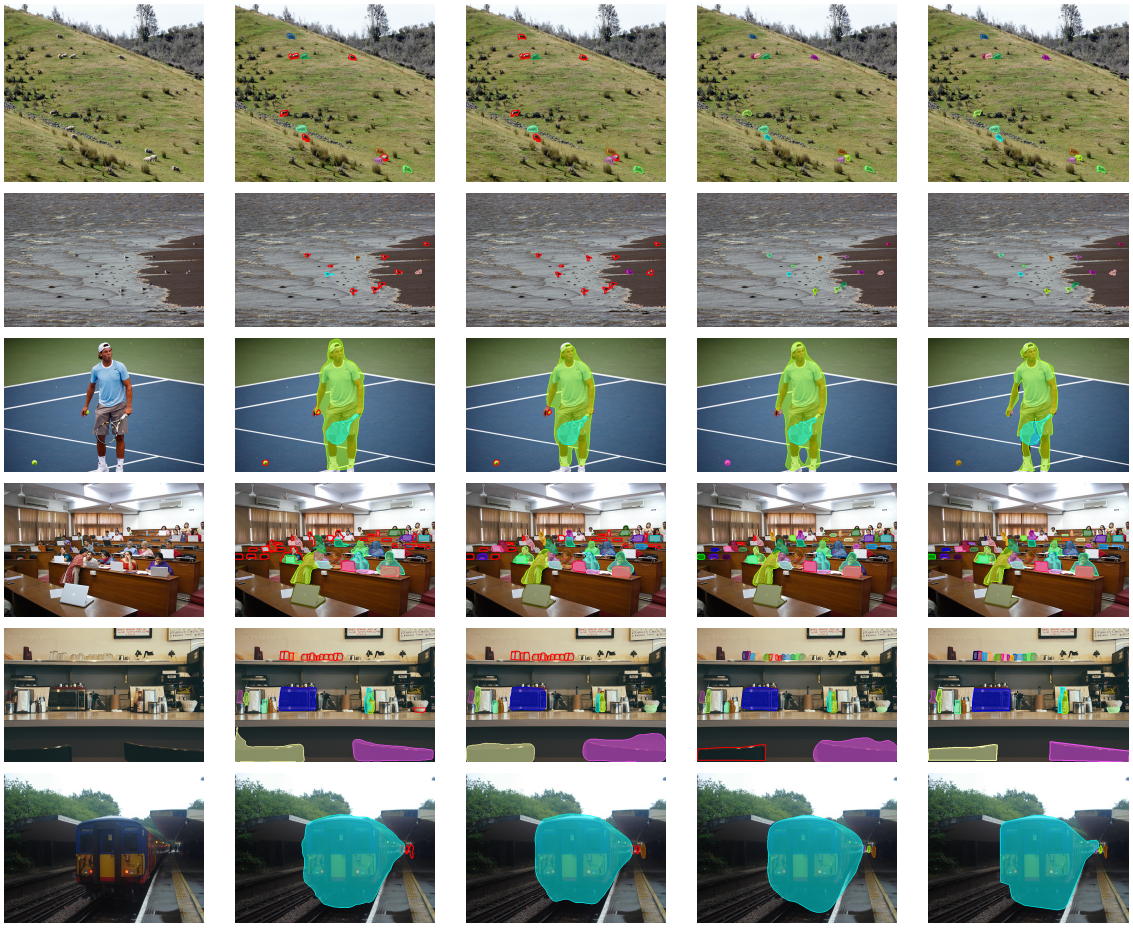


Figure 4.9: Qualitative results of SharpMaskZoom [Pinheiro et al., 2016], FastMask [Hu et al., 2017a], and AttentionMask$_8^{128}$ on images of the COCO test dataset. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) is visualized per annotated object. Input images and annotations taken from the COCO dataset [Lin et al., 2014].

of the larger chairs in this example. In general, large objects are not an area of concern for AttentionMask as the results in Tab. 4.5 indicate. This observation is supported by the final row, where AttentionMask discovers the small passengers on the platform and the large train.

Across the results in Fig. 4.9, AttentionMask discovers most small objects that FastMask and SharpMaskZoom miss. Despite these strong results, all three presented systems generate object proposals that do not adhere well to the object boundaries. The loosely fitting object proposals for the tennis player in the third row highlight this weak boundary adherence. We will discuss this problem and possible solutions in more detail in Ch. 5 - Ch. 7.

4.4.2 Results with Bounding Box Proposals

The focus of this thesis is on pixel-precise object proposals. Still, we show quantitative results of the evaluation using box proposals on the COCO test set in Tab. 4.6. To generate a box proposal from a mask proposal, we generate a bounding box around the mask proposal.

Table 4.6: Results on the COCO test dataset using bounding box proposals in terms of six Average Recall (AR) measures and the runtime. AR^S, AR^M, and AR^L denote results on small, medium, and large objects. See Tab. 2.1 for details on the AR variations. For mask-based proposals, the bounding box per proposal is used. **Bold font** highlights the best results, while *italic font* indicates the second-best results. †: Not including generation of edge maps. *: GPU-only runtime.

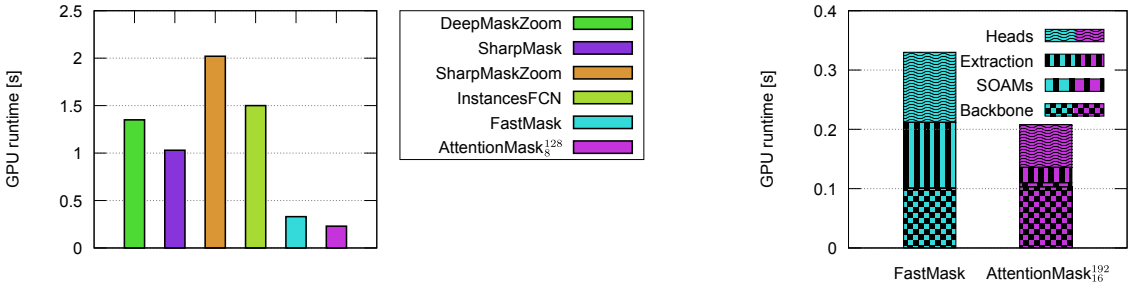
Method	AR@10↑	AR@100↑	AR@1k↑	AR^S@100↑	AR^M@100↑	AR^L@100↑	Runtime↓
BING	0.037	0.084	0.163	-	-	-	**0.20**s
Edge Boxes	0.074	0.178	0.338	0.017	0.138	0.505	0.31s
MCG	0.101	0.246	0.398	0.051	0.232	0.591	45s
COB	0.127	0.278	0.462	0.077	0.280	0.615	45s
DeepMaskZoom	0.191	0.378	0.511	0.141	0.493	0.617	1.35s*
SharpMask	0.198	0.367	0.490	0.063	0.514	0.674	1.03s*
SharpMaskZoom	0.202	0.397	0.533	0.147	0.519	0.648	2.02s*
FastMask	**0.227**	*0.430*	0.568	0.175	**0.549**	0.692	0.33s*
$\text{AttentionMask}_8^{128}$	0.214	0.426	*0.570*	**0.210**	0.508	0.673	0.22s*
$\text{AttentionMask}_8^{192}$	*0.221*	**0.435**	**0.576**	*0.206*	0.512	*0.710*	0.22s*
$\text{AttentionMask}_{16}^{192}$	0.219	0.425	0.554	0.148	*0.542*	**0.726**	*0.21*s*

Overall, AttentionMask shows a good performance on the complex COCO test dataset. $\text{AttentionMask}_8^{192}$ outperforms all other methods in terms of AR@100 and AR@1000 while trailing only FastMask in AR@10. Similar to the results based on masks, $\text{AttentionMask}_8^{128}$ and $\text{AttentionMask}_8^{192}$ outperform all other systems on small objects. However, the improvement from FastMask to AttentionMask is only 20.0% compared to 52.8% in the case of masks. Depending on the AttentionMask version, the results for medium and large objects are slightly above or below FastMask. Compared to the computationally efficient methods BING and Edge Boxes, which do not utilize CNNs, $\text{AttentionMask}_8^{192}$ leads to substantial improvements in terms of AR@100 (+418% and +144%).

4.4.3 Evaluation of the GPU Runtime

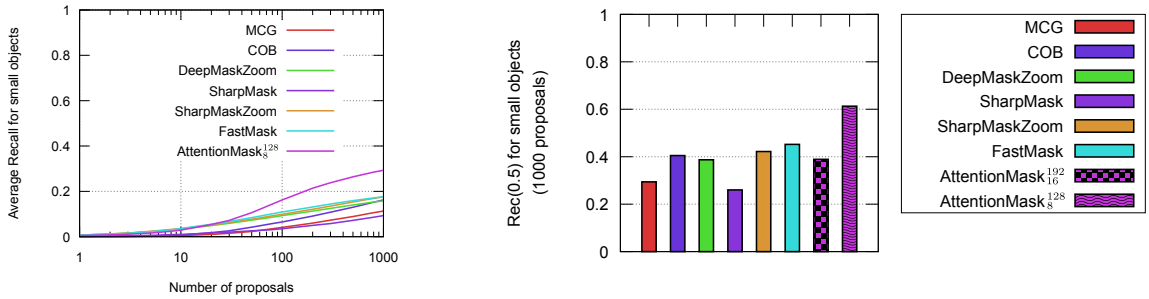
Besides the results for discovering objects, Tab. 4.5 and Tab. 4.6 also present the runtimes for multiple systems. The runtimes indicate that the three AttentionMask versions are the fastest systems evaluated for generating mask proposals. This is also visible from the plot in Fig. 4.10(a), which compares the GPU runtimes of the CNN-based systems. As expected, AttentionMask is considerably faster than the multi-shot approaches DeepMaskZoom, SharpMask, and SharpMaskZoom (up to 90.0% faster). Moreover, $\text{AttentionMask}_8^{128}$ is 33% faster than FastMask while discovering more objects. Similar results are visible for box proposals. Only BING is slightly faster than AttentionMask. However, BING discovers substantially fewer objects.

Furthermore, we investigate which parts of AttentionMask contribute to the improved runtime compared to FastMask. To this end, Fig. 4.10(b) presents the GPU runtimes of different parts inside FastMask and $\text{AttentionMask}_{16}^{192}$, the version of AttentionMask most similar to FastMask. The results show that the runtime for the window extraction is significantly reduced (-76.6%) using the selective window extraction based on the SOAMs. Similarly, the selective window extraction also reduces the runtime for objectness scoring and segmentation by 39.0%. Due to their lightweight design, the SOAMs themselves contribute only slightly to



(a) Comparison of the GPU runtime **(b)** Detailed GPU runtime

Figure 4.10: Detailed analysis of the GPU runtime. (a): Comparison of the GPU runtime across the CNN-based object proposal generation systems from Tab. 4.5. (b): Detailed analysis FastMask's and AttentionMask$_{16}^{192}$'s GPU runtime with contributions of the different system parts. The parts comprise the feature extraction in the backbone and the feature pyramid construction (Backbone), the SOAMs, the window extraction (Extraction), and the segmentation and objectness scoring (Heads).



(a) AR^S **(b)** Rec(0.5) for small objects

Figure 4.11: Detailed analysis of the discovery of small objects. (a): Comparison of different CNN-based object proposal generation approaches in terms of Average Recall for small objects (AR^S) across various numbers of proposals. (b): Recall (Rec(0.5)) of different CNN-based object proposal generation systems for small objects (IoU ≥ 0.5, 1000 proposals). Both results were generated on the COCO test set.

the runtime. Overall, adding the small overhead for the SOAMs, AttentionMask exhibits a significant speedup for the entire system (36.8% faster).

4.4.4 Ablation Studies

To better understand the influence of the new pyramid level $\mathcal{S}_8$ and the SOAMs, we carry out two ablation studies described below.

Influence of the Pyramid Level $\mathcal{S}_8$

The previous results show that the new pyramid level $\mathcal{S}_8$ substantially improves the results on small objects. Figure 4.11(a) presents the results in terms of AR^S for 1 to 1000 proposals on the COCO test dataset to investigate this improvement in more detail. The results indicate that AttentionMask$_8^{128}$ improves the discovery of small objects across various numbers of proposals. As the number of proposals grows, the gap between AttentionMask and the other

(a) Input image **(b)** AttentionMask$_8^{128}$

Figure 4.12: Qualitative result of AttentionMask$_8^{128}$ (b) on an input image from the COCO dataset with 13 tiny bird (a). Despite the generally strong performance on small objects, AttentionMask$_8^{128}$ discovers only one birds. Filled colored results denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) is visualized per annotated object. Input images and annotations taken from the COCO dataset [Lin et al., 2014].

systems increases. This is highlighted by an improvement of 66.4% over FastMask on 1000 proposals. To better assess the number of discovered small objects, Fig. 4.11(b) presents the results in terms of Rec(0.5) on the COCO test set. The results show that AttentionMask$_8^{128}$ utilizing $\mathcal{S}_8$ outperforms all other systems and discovers 61.3% of the small objects. In contrast, FastMask discovers only 45.2% of the small objects. These results again highlight the positive influence of $\mathcal{S}_8$ for the discovery of small objects.

Despite the good overall results on small objects, several small objects are still missed by AttentionMask$_8^{128}$ as the results in Fig. 4.11 indicate. Hence, besides generally difficult object characteristics like an elongated shape or low contrast (see Ch. 8), the small size of objects still poses a challenge. For instance, tiny objects like the birds in Fig. 4.12 are missed by AttentionMask$_8^{128}$. Since the birds are smaller than 80 pixels, they are technically too small for the size condition of $\mathcal{S}_8$ during training. Hence, the network can not learn to discover these objects. We will discuss the effect of tiny objects on the results of object proposal generation methods in more detail in Sec. 8.2.1.

Influence of the Scale-specific Objectness Attention Modules

To conclude the evaluation, we discuss the influence of the proposed SOAMs on the results of AttentionMask. As previously discussed, only the increased efficiency utilizing the SOAMs allows the addition of $\mathcal{S}_8$ and the strong results on small objects. Furthermore, the SOAMs reduce the number of false positives as the comparison between FastMask and AttentionMask$_{16}^{192}$ in Tab. 4.5 indicates (+ 6.8% in AR@100). The main difference between the two systems is the addition of the SOAMs in AttentionMask$_{16}^{192}$. To investigate this effect in more detail, we determine how many windows were correctly recalled by the SOAMs and how many windows were pruned. The results in Tab. 4.7 show that the SOAMs prune the majority of windows. On $\mathcal{S}_8$, which contributes the majority of all windows, 93.9% of the windows are not extracted. Nevertheless, more than 78% of the positive window samples are recalled for $\mathcal{S}_8$. Across the pyramid levels $\mathcal{S}_{32}, \mathcal{S}_{64}$, and $\mathcal{S}_{128}$, the recall is even above 90%. Hence, pruning windows based on the SOAMs leads to few missed objects while removing many possible false positives and

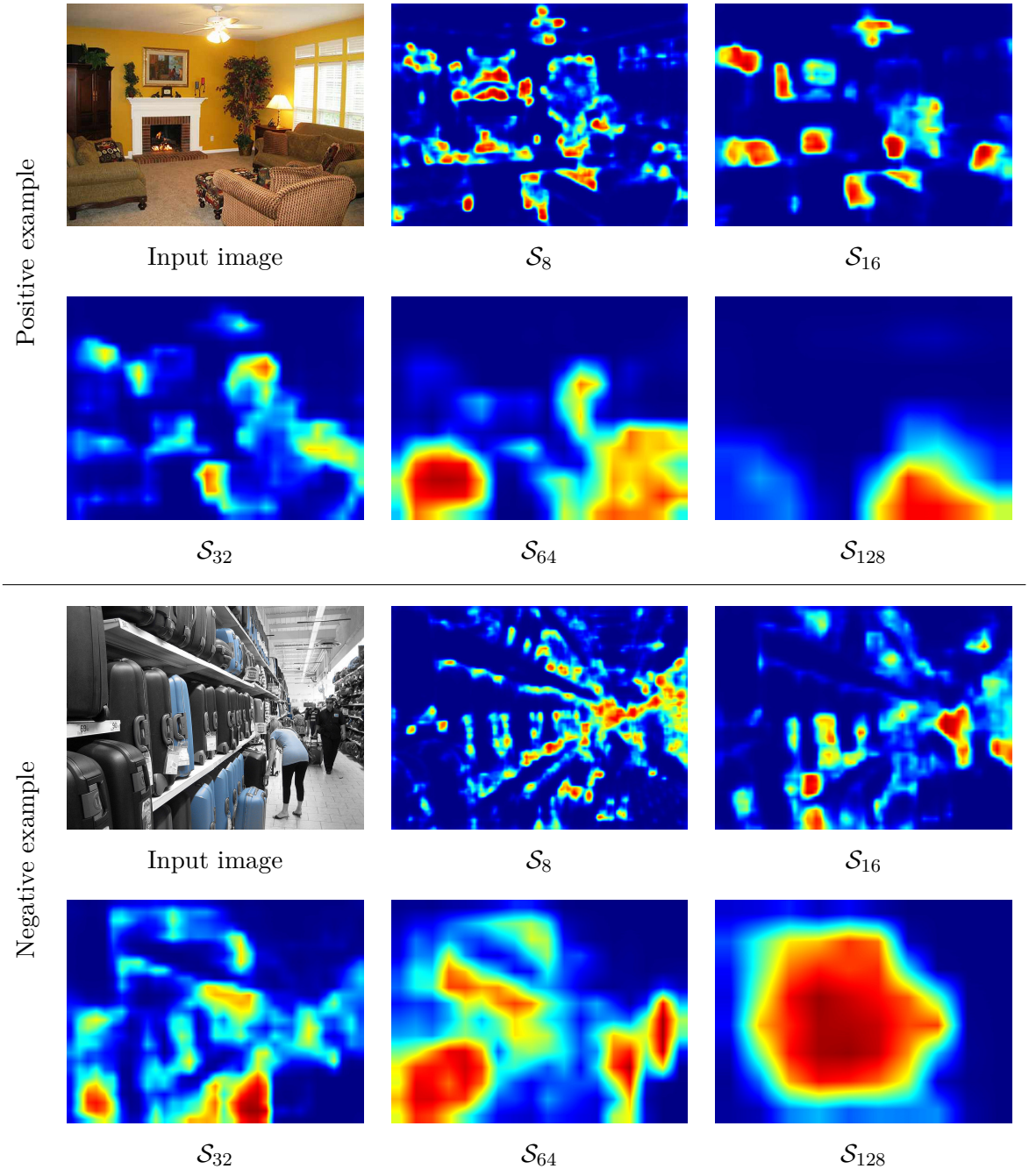


Figure 4.13: Scale-specific objectness attention maps in AttentionMask$_8^{128}$ for two input images from the COCO dataset. The attention maps are extracted from the pyramid levels $\mathcal{S}_8, \mathcal{S}_{16}, \mathcal{S}_{32}, \mathcal{S}_{64}$, and $\mathcal{S}_{128}$, highlighting small to large objects. The scale-specific objectness attention maps in the upper examples strongly focus on objects or object parts. In contrast, the scale-specific objectness attention maps in the lower example do not properly cover the suitcases. All scale-specific objectness attention maps are upsampled to input image size for improved visibility. Red areas indicate high attention, while blue areas indicate low attention. Input images taken from the COCO dataset [Lin et al., 2014].

Table 4.7: Detailed analysis of the SOAMs' effect on the window extraction per pyramid level. The window recall measures the relative number of extracted windows from all positive samples, while pruned windows denotes the relative number of windows pruned given all possible windows. The results were generated on the COCO test set.

Pyramid level	Windows recalled↑	Windows pruned↑
$\mathcal{S}_8$	0.781	0.939
$\mathcal{S}_{16}$	0.831	0.924
$\mathcal{S}_{32}$	0.982	0.640
$\mathcal{S}_{64}$	0.969	0.594
$\mathcal{S}_{128}$	0.918	0.704

allowing better utilization of the limited GPU resources to improve the discovery of small objects.

To investigate the strengths and the scale-specific nature of the SOAMs, we already presented sample outputs in Fig. 4.5. Figure 4.13 presents additional scale-specific objectness attention maps, the results of the SOAMs, on more cluttered scenes. These qualitative results generally indicate that the SOAMs capture most objects on the correct pyramid levels. For instance, in the upper example, the first scale-specific objectness attention map ($\mathcal{S}_8$) covers the small objects on the fireplace or object parts, while the final two maps ($\mathcal{S}_{64}$ and $\mathcal{S}_{128}$) cover the armchairs and the sofa. Hence, the SOAMs are not distracted by the clutter in the image. The lower example in Fig. 4.13 shows mixed results. While the people are properly captured on $\mathcal{S}_{16}$ to $\mathcal{S}_{64}$, most suitcases on the shelf are not highlighted at the correct pyramid level. Although the final scale-specific objectness attention map ($\mathcal{S}_{128}$) covers the entire shelf, discovering the individual medium suitcases is hardly possible from $\mathcal{S}_{128}$ due to the size mismatch. Overall, few examples for missed objects or size mismatches exist, while most objects and object parts are properly highlighted by our scale-specific objectness attention maps.

4.5 Discussion

In this chapter, we proposed our novel CNN-based object proposal generation system AttentionMask. AttentionMask follows the one-shot concept and extracts windows from an internal feature pyramid representing the input image at multiple scales to discover objects. Unlike previous systems, AttentionMask utilizes our new scale-specific objectness attention to focus the window extraction on relevant parts of the feature pyramid. This focused, attention-based processing increases the computational efficiency and frees up GPU resources for a more detailed analysis of small objects. To this end, we add a new pyramid level $\mathcal{S}_8$ to the base of the feature pyramid to improve the discovery of small objects. Without our more efficient processing pipeline utilizing attention, the addition of $\mathcal{S}_8$ and the improved results on small objects would not be possible. As a result, AttentionMask addresses a major limitation of object proposal generation systems discussed in the introduction, the discovery of small objects.

Our detailed evaluation showed that AttentionMask outperforms all mask-based state-of-the-art object proposal generation systems on the complex COCO dataset. On the very challenging small objects, AttentionMask outperforms state-of-the-art systems by at least 52.8% while the improvements across all object sizes are at least 7.9%. These improvements

showcase the benefit of our new scale-specific objectness attention and the additional pyramid level $\mathcal{S}_8$. Moreover, the focused processing based on the scale-specific objectness attention leads to the fastest runtime among mask-based state-of-the-art object proposal generation methods.

Despite the improved results on small objects, some limitations remain. First, discovering tiny objects covering only a few pixels is still challenging due to the downsampling process inside the CNN. Second, the generated proposals do not adhere well to the object boundaries, similar to other CNN-based systems. As mentioned in the introduction, this is one of the major limitations of CNN-based object proposal generation systems. Finally, the ranking of the object proposals is suboptimal as discussed in the introduction. This limitation is highlighted by the difference in AR between the first 10 proposals and the first 1000 proposals across all systems.

Overall, this chapter introduced our novel and efficient object proposal generation system AttentionMask that outperforms all previous systems while reducing the runtime and explicitly addressing the limitation of discovering small objects. Some of the remaining limitations like the coarse object proposals will be tackled in the subsequent three chapters. Applications of AttentionMask in complex, real-world scenarios will be presented in Ch. 9.

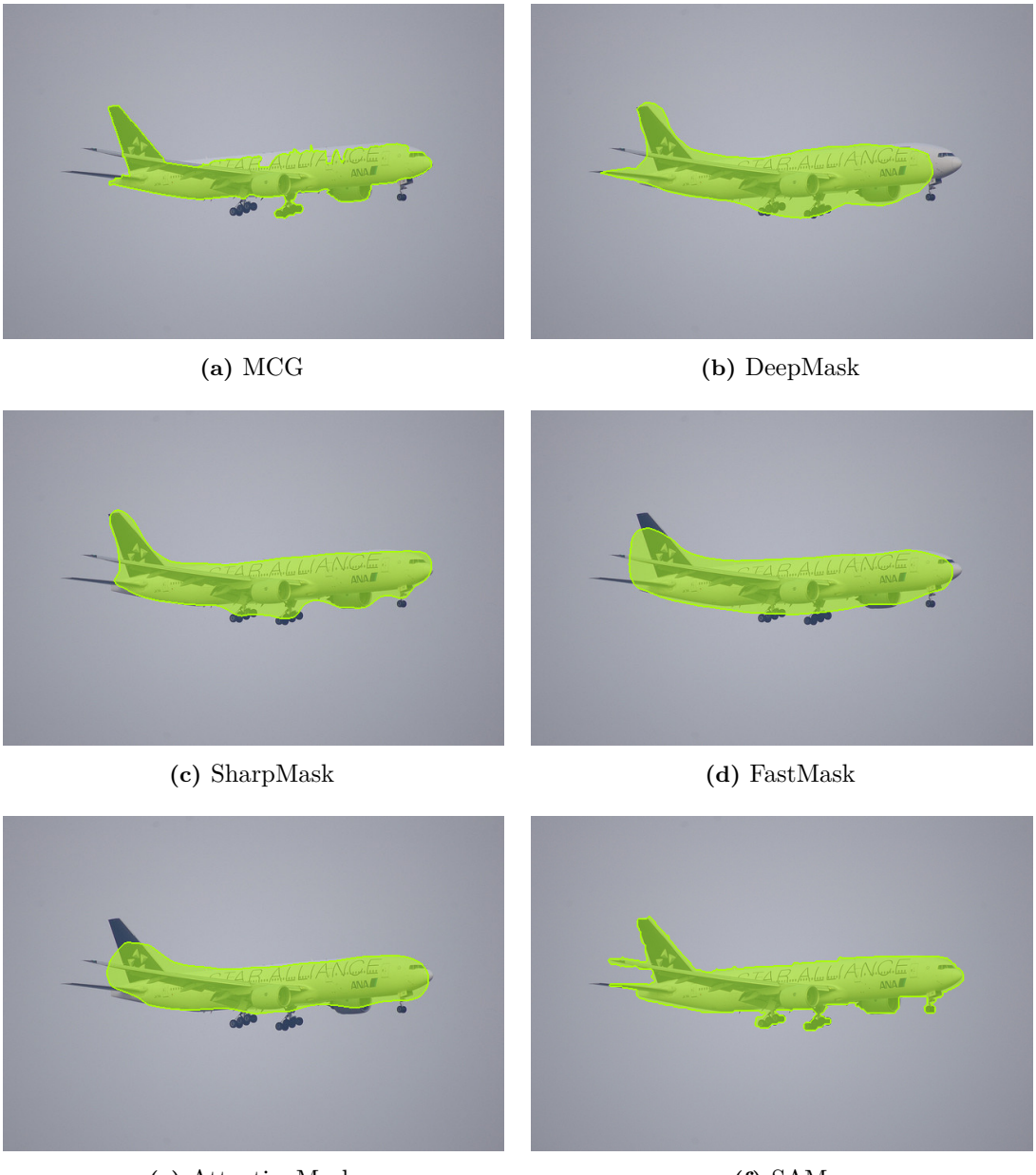
Chapter 5

SAM: Superpixel-based Refinement for Object Proposals

Table of Contents

The previous chapter introduced our efficient object proposal generation system AttentionMask. Despite significant improvements in terms of Average Recall (AR), AttentionMask mostly generates coarse proposals. A typical example of this is visible in Fig. 5.1(e). Although the AttentionMask proposal discovers the airplane, the segmentation mask does not adhere precisely to the object boundaries. The issue exists in all CNN-based systems and can be considered as a trade-off between CNN-based systems [Pinheiro et al., 2015, 2016; Hu et al., 2017a] and traditional systems that do not utilize CNNs [Uijlings et al., 2013; Krähenbühl and Koltun, 2014; Pont-Tuset et al., 2017]. While the CNN-based systems capture most objects entirely, the traditional systems generate precise proposals but miss more objects or object parts (see Fig. 5.1). We already presented this trade-off in the introduction as one of the major limitations in object proposal generation.

The main reason behind the coarse proposals in CNN-based systems is the downsampling process in the backbone CNN that removes spatial information necessary to generate precise proposals. However, this downsampling process is integral in CNNs to generate semantically rich features without exceeding memory limitations. For instance, in DeepMask [Pinheiro et al., 2015], the input image is downsampled four times in the backbone to generate a feature map that is downsampled by a factor of 16 w.r.t. the input image. The approaches following the efficient one-shot concept suffer even more from this problem due to the additional



(a) MCG **(b)** DeepMask

(c) SharpMask **(d)** FastMask

(e) AttentionMask **(f)** SAM

Figure 5.1: Results of the five object proposal generation systems MCG [Pont-Tuset et al., 2017] (a), DeepMask [Pinheiro et al., 2015] (b), SharpMask [Pinheiro et al., 2016] (c), FastMask [Hu et al., 2017a] (d), and our previously introduced AttentionMask (e). Moreover, (f) shows the result of our new object proposal generation system Superpixel-based AttentionMask (SAM). The proposal produced by MCG adheres well to the object boundaries while missing parts of the airplane. The proposals by the CNN-based systems DeepMask, SharpMask, FastMask, and AttentionMask discover the entire object but only loosely fit the object boundaries. Applying our superpixel-based refinement as part of SAM leads to a proposal discovering the entire airplane and precisely segmenting it, including fine details like the landing gear or the wings tips. Input image taken from LVIS dataset [Gupta et al., 2019].

downsampling in the internal feature pyramid. As a result, in FastMask [Hu et al., 2017a] and AttentionMask, all segmentation masks are generated based on 10×10 feature maps that are upsampled as described in Sec. 3.2.3 and Sec. 4.2.4. Consequently, small objects and large objects are segmented based on feature maps of the same spatial resolution, which leads to coarse, imprecise proposals for larger objects.

A common solution to this problem would be using an encoder-decoder architecture like in semantic segmentation [Lin et al., 2017a; Chen et al., 2017] or salient object detection [Li and Yu, 2016; Qin et al., 2019]. The decoder guides the upsampling of semantically rich but coarse features with higher resolution features from earlier network layers in those systems. In object proposal generation, Pinheiro et al. [2016] utilize this approach in SharpMask. However, SharpMask uses the computationally inefficient multi-shot concept. Additionally, the refined proposals are still imprecise as visible around the landing gear and airplane's nose in Fig. 5.1(c).

Conditional Random Fields (CRFs) are another approach for refining coarse results that is commonly used in semantic segmentation [Chen et al., 2017; Chan et al., 2019] and salient object detection [Li and Yu, 2016; Hou et al., 2017]. CRFs typically utilize low-level information like color at input image resolution to post-process coarse initial results [Li and Yu, 2016; Chen et al., 2017; Chan et al., 2019]. However, different from semantic segmentation or salient object detection, hundreds of proposals per image need refinement in object proposal generation. Since CRF-based post-processing is computationally intensive [Ke et al., 2018], an application to hundreds of proposals is infeasible. Other approaches to improve low-resolution results use dilated convolutions [Yu and Koltun, 2016; Chen et al., 2017] or iterative refinements [Zhang et al., 2019a; Kirillov et al., 2020]. Still, both concepts are computationally demanding if applied to large networks or hundreds of proposals per image. Therefore, a method is needed that refines the object proposals while sharing computation between the proposals of an image to increase efficiency.

Inspired by earlier work on object proposal generation not utilizing CNNs [Uijlings et al., 2013; Krähenbühl and Koltun, 2014; Pont-Tuset et al., 2017], we propose to use superpixels [Ren and Malik, 2003] introduced in Sec. 2.1.1 for refining coarse initial proposals. Superpixels are useful for precisely segmenting objects since they can be shared between the proposals of an image to increase efficiency and capture complex object shapes while reducing the number of basic entities in an image. Moreover, they can be shared between the proposals of an image to increase efficiency. For instance, MCG [Pont-Tuset et al., 2017] generates precise object proposals using superpixel segmentations. However, those proposals miss several objects or object parts as visible from the result in Fig. 5.1(a).

This leads to our new object proposal generation system *Superpixel-based AttentionMask* (SAM). In SAM, the coarse AttentionMask proposals roughly discover entire objects, while a superpixel-based refinement utilizes precise superpixels to increase the boundary adherence of the initial, coarse proposals. Hence, SAM presents an innovative combination of traditional superpixel-based and modern CNN-based object proposal generation approaches. To combine CNNs with superpixels, we aggregate features across superpixels in a superpixel pooling framework [Mostajabi et al., 2015; He et al., 2017b; Kwak et al., 2017; Park et al., 2017]. Starting from the coarse AttentionMask proposals, SAM generates precise superpixel segmentations and extracts CNN features from AttentionMask's backbone as visible in Fig. 5.2. Using the superpixels, SAM pools a feature vector for each superpixel based on the coarse proposal and learned features in our superpixel-based refinement module (see Fig. 5.2). Subsequently, the superpixels within and around each coarse proposal are classified as part of the object proposal

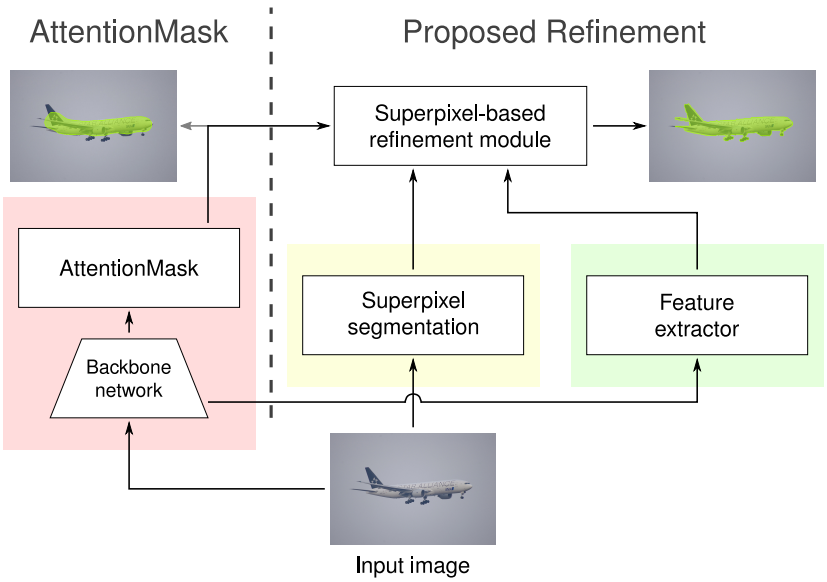


Figure 5.2: Visualization of the core building blocks of our object proposal generation system *Superpixel-based AttentionMask* (SAM). SAM starts by applying AttentionMask (red box) introduced in Ch. 4 to generate coarse object proposals. In parallel, superpixel segmentations are generated (yellow box), and features are extracted from AttentionMask's backbone network (green box). Our novel superpixel-based refinement module (top) combines the three streams applying superpixel pooling and superpixel classification. This leads to refined proposals (top right) based on coarse AttentionMask results (top left). Input image taken from the LVIS dataset [Gupta et al., 2019].

or the background. This superpixel classification yields the refined proposals that precisely capture entire objects by combining CNN and superpixel information.

In this chapter, we present or novel object proposal generation system SAM based on our publications Wilms and Frintrop [2020] as well as Wilms and Frintrop [2021]. First, in Sec. 5.1 we review relevant literature on the non-trivial integration of superpixels into CNNs. Second, we discuss the innovative architecture of SAM utilizing superpixel pooling and a superpixel classifier within our novel superpixel-based refinement module in Sec. 5.2. Additionally, we present a four-stage post-processing to further improve the results. Subsequently, Sec. 5.3 presents the end-to-end training of SAM. In Sec. 5.4, we evaluate SAM on the challenging LVIS dataset [Gupta et al., 2019] and discuss the effects of the superpixel-based refinement. Additionally, ablation studies present insights on the choice of the superpixel segmentation method for this task. We discuss the contributions and limitations of SAM as well as major findings in Sec. 5.5 to conclude the chapter.

5.1 Superpixels in CNNs

The combination of superpixels and CNNs is non-trivial. Superpixels are a precise representation of the image content capturing arbitrarily complex shapes. The arbitrary shapes can lead to a highly irregular topology of the superpixel segmentation[1] with a varying number of neighbors per superpixel. Moreover, the relative spatial locations between neighboring

[1]The superpixel segmentation methods by Moore et al. [2008] and Moore et al. [2010] enforce a regular grid-topology, but lack segmentation quality.

superpixels are not fixed. In contrast, images have a regular grid topology with four or eight neighbors per pixel, except for the image borders. Given the lack of a regular topology and missing relative location information in superpixel segmentations, CNNs that apply $n \times n$ kernels can not directly work on superpixels.

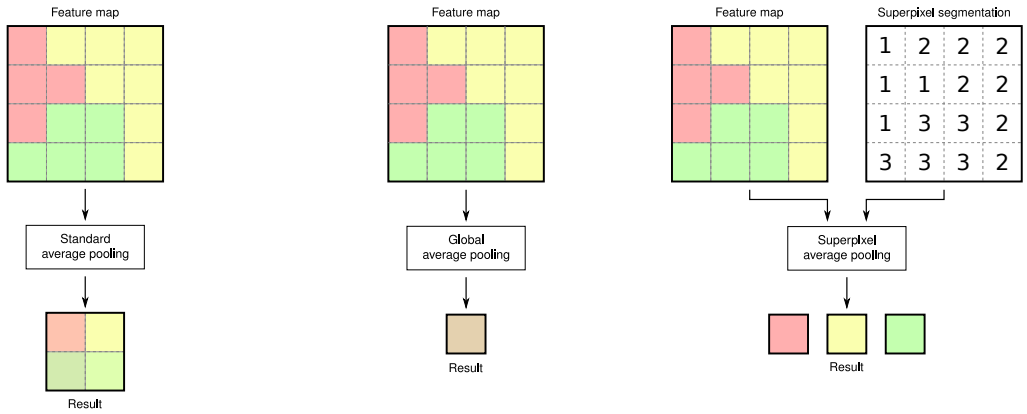
Few approaches have been presented to circumvent this problem and integrate superpixels into CNNs. He et al. [2015] change the input structure by extracting a feature vector per superpixel and calculating the contrast between all superpixels in the image. Subsequently, they apply a CNN with 1D-convolutions on the results in a salient object detection framework. However, the contrast-based representation focuses on the salient object detection task. Inspired by graph convolutional networks [Kipf and Welling, 2017], Suzuki et al. [2018] propose superpixel convolutions based on the adjacency matrix of the superpixel segmentation. For the rows of the adjacency matrix, a weight vector is learned as a circulant matrix. Subsequently, the convolution is applied on the adjacency matrix, and the results are restored as a pixel-wise feature map. Although this approach integrates superpixels into CNNs, it loses all spatial information between them and prefers a regular superpixel topology.

A different class of approaches uses windows around superpixels to classify superpixels with a CNN in tasks like salient object detection [Zhao et al., 2015; Tang and Wu, 2016; Lee et al., 2016], depth estimation [Liu et al., 2015] or semantic segmentation [Zhao et al., 2017]. Using windows rather than the superpixel itself is beneficial since the input shape of CNNs is rectangular. However, the window-based processing removes all detailed segmentation information for feature extraction inside CNNs.

Chen et al. [2016b] and Hu et al. [2017b] use superpixels to post-process initial blurry pixel-precise results outside the CNN. Different to those approaches, Gadde et al. [2016] propose a Gaussian superpixel-based bilateral filtering at arbitrary intermediate steps of a CNN. Hence, the Gaussian bilateral filtering is applied on features inside the CNN rather than the network's results. The filtering is based on superpixels and their color information to enforce similar features in visually similar and proximate superpixels.

Another line of work uses superpixel pooling for superpixel-based feature aggregation in semantic segmentation [Mostajabi et al., 2015; Kwak et al., 2017; He et al., 2017b; Park et al., 2017]. Superpixel pooling generates a feature vector for each superpixel based on a feature map and a superpixel segmentation. The feature vector represents the average, the maximum, or a random pixel per feature across the superpixel. Thus, the output of the superpixel pooling is a feature vector per superpixel, which will be further processed by the network. This makes superpixel pooling an intermediate concept between standard $n \times n$ pooling and global pooling as Fig. 5.3 demonstrates. In contrast to global pooling, superpixel pooling distinguishes between different image areas. However, these areas are not defined by fixed $n \times n$ windows as in standard pooling but follow superpixels and allow arbitrary shapes. This makes superpixel pooling a flexible alternative to the other pooling strategies. Similar to other pooling strategies, backpropagation through superpixel pooling is straightforward. Overall, superpixel pooling allows an end-to-end integration of superpixels into CNNs that is not limited to the input level or the level of predictions.

In general, four streams of integrating superpixels into CNNs exist. First, the CNN structure is changed to work directly on superpixels, leading to a loss of spatial information early in the network. Second, superpixels are used to extract windows as inputs for the CNN, which removes detailed segmentation information. Third, superpixels are used to enhance dense CNN predictions without end-to-end integration. Finally, the feature aggregation by



(a) Standard average pooling **(b)** Global average pooling **(c)** Superpixel average pooling

Figure 5.3: Three variations of average pooling on a 4×4 feature map with RGB values. Since the pooling grid is fixed, the standard 2×2 average pooling (a) with stride 2 leads to mixing the RGB values. The global average pooling (b) extracts the average per color channel across the entire image, losing all spatial information. As an intermediate concept, superpixel average pooling (c) extracts the average color per superpixel. Given a high-quality superpixel segmentation, the regions are not mixed, and the three resulting elements contain only superpixel-specific information.

superpixel pooling generates a feature representation of each superpixel and allows end-to-end processing of these features. Hence, the superpixel pooling enables the most native integration of superpixels into arbitrary stages of CNNs without loosing spatial information or segmentation details early in the network. Note that after the superpixels pooling, the spatial information of the superpixels might be lost. However, this only happens after initial feature generation in the CNN backbone.

5.2 Superpixel-based AttentionMask

This section presents the architecture of our novel object proposal generation system *Superpixel-based AttentionMask* (SAM) that refines coarse object proposals utilizing superpixels in a superpixel pooling framework. Figure 5.4 shows an overview of SAM that consists of two main parts. First, three base streams (red, yellow, and green blocks in Fig. 5.4) generate coarse AttentionMask proposals (red block), precise superpixel segmentations (yellow block), and per-pixel features from AttentionMask's backbone (green block). As described in Sec. 5.2.1, the streams are synchronized by the SOAMs that guide the window extraction across the pyramids and stacks. The second part of SAM, described in Sec. 5.2.2, consists of our novel superpixel-based refinement module. This module refines each coarse AttentionMask proposal utilizing the precise superpixel segmentations and the extracted features. In this innovative process, we utilize superpixel average pooling twice to generate a superpixelized version of the upsampled coarse proposal and extract per superpixel features. Based on this information, a novel superpixel classifier assigns the superpixels to the object proposal or the background. This leads to highly precise object proposals generated by SAM, which are further enhanced using a four-stage post-processing described in Sec. 5.2.3.

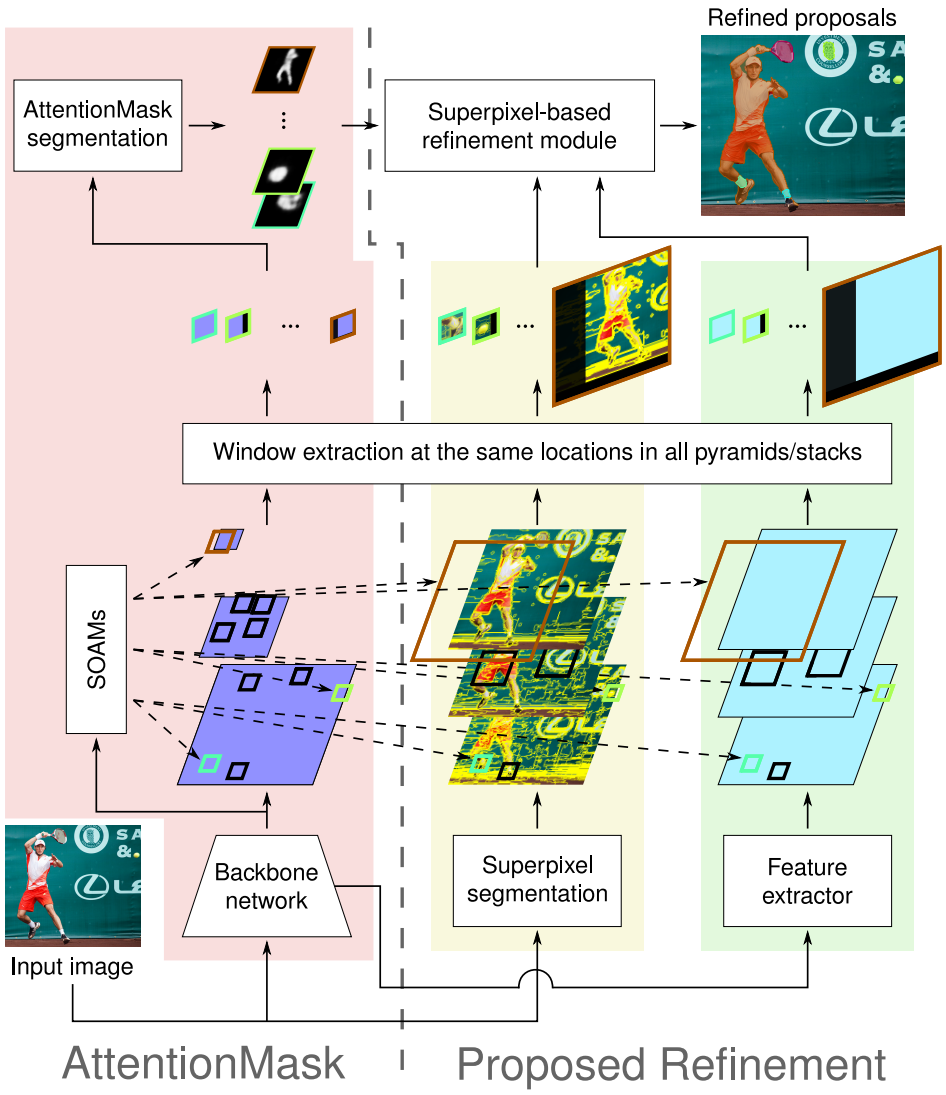


Figure 5.4: Overview of our object proposal generation system SAM. SAM consists of three base streams for generating coarse AttentionMask object proposals (red block), superpixel segmentations (yellow block), and learned features (green block). For each coarse proposal, the respective windows are extracted from the other streams based on the SOAM outputs (see colored windows across the streams). Finally, the three streams are combined in our novel superpixel-based refinement module utilizing superpixel pooling and a superpixel classifier. This results in refined proposals that better adhere to the object boundaries. Figure 5.5 presents a more detailed visualization of our superpixel-based refinement module. Input image taken from the LVIS dataset [Gupta et al., 2019].

Table 5.1: Number of superpixels per superpixel segmentation for each of AttentionMask's feature pyramid levels in SAM. Since $\mathcal{S}_8$ and $\mathcal{S}_{16}$ capture small objects, more superpixels are used and vice versa.

Pyramid level	$\mathcal{S}_8$	$\mathcal{S}_{16}$	$\mathcal{S}_{24}$	$\mathcal{S}_{32}$	$\mathcal{S}_{48}$	$\mathcal{S}_{64}$	$\mathcal{S}_{96}$	$\mathcal{S}_{128}$
Number of superpixels	8000	4000	3000	2000	1500	1000	750	500

5.2.1 Base Streams

SAM is based on three streams as visualized in Fig. 5.4. The first stream (red box in Fig. 5.4) consists of AttentionMask, including the feature pyramid, and generates coarse object proposals as described in Ch. 4. Different to Ch. 4, SAM uses AttentionMask with the *conv1-conv4* stages of the smaller ResNet-34 [He et al., 2016a] as backbone to save GPU memory for the subsequent refinement. The change of the backbone network leads to a slight drop in performance as we will show in Sec. 5.4.1.

The second base stream in SAM (yellow box in Fig. 5.4) creates precise superpixel segmentations of the input image for each pyramid level of AttentionMask's feature pyramid. All superpixel segmentations are generated on input image resolution for a precise refinement yielding a stack of superpixel segmentations rather than a pyramid. The number of superpixels varies from 8000 for the pyramid level $\mathcal{S}_8$ to 500 for the pyramid level $\mathcal{S}_{128}$. The intermediate levels are segmented accordingly with different numbers of superpixels as Tab. 5.1 summarizes. Using different numbers of superpixels per pyramid level is motivated by the size of the objects that fit the different pyramid levels. More superpixels increase the chance of capturing small objects, while fewer superpixels lead to fewer subsequent classifications for larger objects. As superpixel segmentation method, we choose the Felzenszwalb and Huttenlocher method (FH) [Felzenszwalb and Huttenlocher, 2004] described in Sec. 3.1.2. FH produces less oversegmentation than other methods and is utilized in several object proposal generation systems [Alexe et al., 2010; van de Sande et al., 2011; Frintrop et al., 2014]. We will show the benefit of using FH with SAM in Sec. 5.4.3.

As a third stream, SAM contains a feature extractor (green box in Fig. 5.4). The extractor creates a feature representation from the input image for each of AttentionMask's pyramid levels. These feature representations (turquoise boxes in Fig. 5.4) yield a stack and serve as the base for pooling superpixel features in the superpixel-based refinement module. To generate features per pyramid level, SAM applies a 1×1 convolution with 256 kernels to the final feature map of the *conv2* stage from AttentionMask's backbone. Subsequently, the features are upsampled to match the resolution of the superpixel segmentations. Utilizing features from the *conv2* stage of the backbone is beneficial compared to other stages as the results in Tab. 5.2 indicate.

The three streams of SAM are linked by the SOAMs from AttentionMask. The SOAMs highlight areas with objects of relevant size per feature pyramid level and focus the extraction of windows on these areas. From the feature pyramid, fixed-size 10×10 windows are extracted in SAM as described in Sec. 4.2.2 for AttentionMask. For the superpixel segmentation and feature extraction streams, the SOAMs are utilized as well to synchronize the extraction. However, instead of extracting fixed-size 10×10 windows, the window size is adapted to the input image resolution. Hence, a 10×10 window in pyramid level $\mathcal{S}_8$ leads to an 80×80 window for the superpixel segmentation or the feature extraction stream (e.g., brown frames in Fig. 5.4). This upsampling is necessary to extract the same area across all three streams

Table 5.2: Comparison of the feature extraction from different stages of the ResNet-34 backbone in SAM. The stages are described in the first column. AR denotes the Average Recall for the first 10, 100, or 1000 proposals on the LVIS validation dataset. We use SAM with only five pyramid levels and the features from the respective stage. The chosen architecture is highlighted in **bold font**.

ResNet stage	AR@10↑	AR@100↑	AR@1000↑
conv1	0.101	0.219	0.315
conv2	**0.100**	**0.221**	**0.320**
conv3	0.099	0.219	0.318
conv4	0.100	0.219	0.317

and allows to preserve spatial details in the superpixel segmentation and feature extraction streams.

Overall, for each coarse AttentionMask proposal of size 10×10, SAM extracts a window from the respective superpixel segmentation and the respective feature map. The superpixel segmentation and feature map windows represent the 10×10 proposals on input image resolution to preserve spatial details.

5.2.2 Superpixel-based Refinement Module

Based on the windows of the three base streams, our novel superpixel-based refinement module connects the three streams and creates a refined proposal. This process is visualized in more detail in Fig. 5.5. First, we upsample the coarse 10×10 AttentionMask proposal to input image resolution. The upsampling leads to a blurry proposal as visible in the top left in Fig. 5.5. To recover the sharp boundaries, we apply superpixel average pooling to the blurry proposal based on the superpixel segmentation window. This leads to a superpixelized version of the proposal with sharp boundaries called *mask prior* (top center in Fig. 5.5). The mask prior contains the average proposal value[2] across each superpixel and already gives a good approximation of the object boundaries.

Instead of directly binarizing the mask prior to generate the refined proposal, we augment the mask prior with semantically rich learned features utilizing the extracted feature map window (bottom right in Fig. 5.5). Similar to the mask prior, we apply superpixel average pooling on the feature map window with the same superpixel segmentation as before. Hence, we create a 256D feature vector for each superpixel in the mask prior (bottom center in Fig. 5.5). In contrast to the previous superpixel pooling on the upsampled proposal, this second superpixel pooling is technically shared between all proposals of a pyramid level, since the features maps and the superpixel segmentations do not change. To combine the mask prior and the feature vector, we use simple concatenation per superpixel (top center in Fig. 5.5). The concatenation is favorable compared to weighting the features with the respective mask prior values, as the results in Tab. 5.3 indicate. The concatenation leads to the final 257D feature vector representation for each superpixel.

[2]The values of the coarse object proposal represent the initial likelihood of a pixel belonging to the object according to AttentionMask.

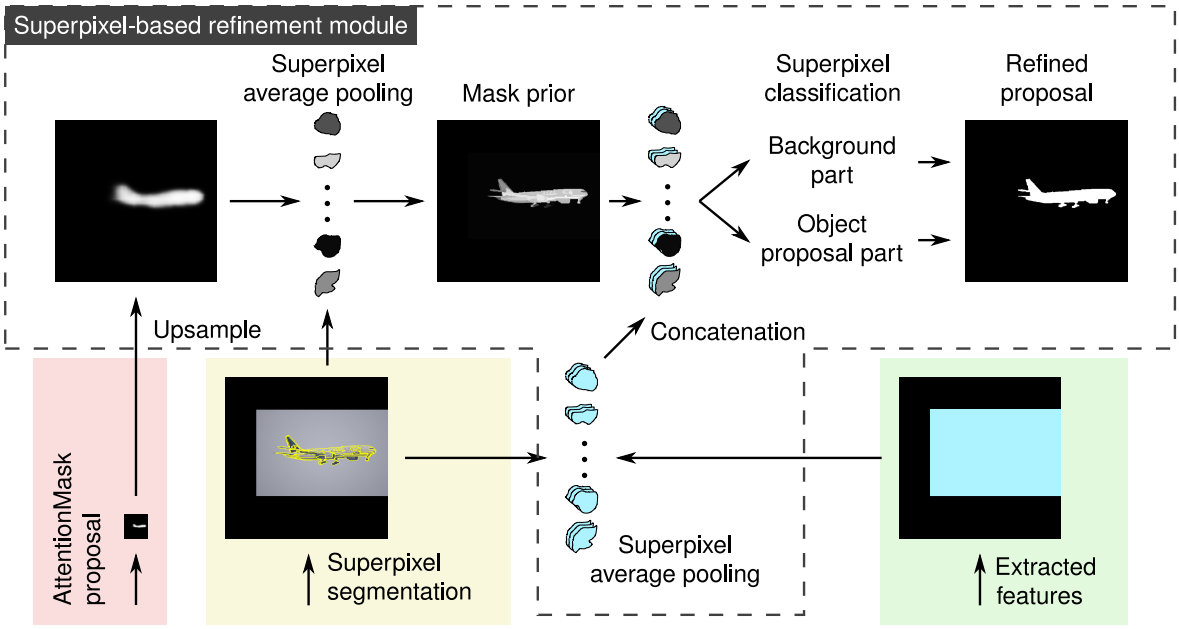


Figure 5.5: Detailed visualization of the proposed superpixel-based refinement module in SAM using the outputs of the three streams (red, yellow, and green) from Fig. 5.4. The coarse AttentionMask proposal is upsampled and superpixelized using superpixel pooling to eliminate the blurred and imprecise boundaries (top left). Subsequently, a feature vector per superpixel is generated using superpixel pooling on the learned features extracted from AttentionMask's backbone (bottom center). The mask prior and the extracted features are combined using concatenated (top center). Finally, our superpixel classifier assigns each superpixel to the object proposal or the background, yielding a precise object proposal (top right). Input image taken from the LVIS dataset [Gupta et al., 2019].

Table 5.3: Comparison of concatenation and weighting for fusing the mask prior and the learned superpixel feature vector in SAM. The strategies are described in the first column. AR denotes the Average Recall for the first 10, 100, or 1000 proposals on the LVIS validation dataset. We use SAM with only five pyramid levels and apply the respective fusion strategy in the superpixel-based refinement module. The chosen strategy is highlighted in **bold font**.

Strategy	AR@10↑	AR@100↑	AR@1000↑
Concatenation	**0.100**	**0.221**	**0.320**
Weighting	0.039	0.061	0.097

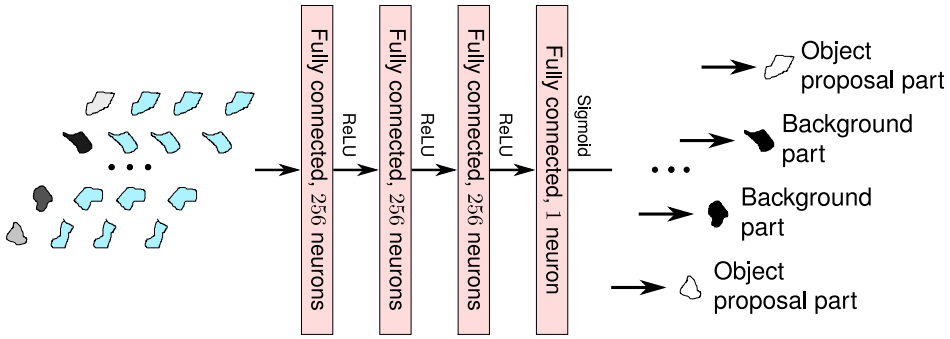


Figure 5.6: Architecture of the proposed superpixel classifier in SAM. The input is a batch of superpixels, where each superpixel is represented by the mask prior (grayish superpixel) and the learned features (turquoise superpixels). The four-layer classifier assigns each superpixel individually to the object proposal (white) or the background (black).

Table 5.4: Comparison of architectures for the proposed superpixel classifier in SAM. The first column describes the architecture in terms of the number of neurons per fully connected layer. We compare models with three to five layers and up to 512 neurons per layer. The final layer has 1 neuron in all architectures corresponding to the binary classification result. AR denotes the Average Recall for the first 10, 100, or 1000 proposals on the LVIS validation dataset. We use SAM with only five pyramid levels and the respective architecture for the superpixel classifier. The chosen architecture is highlighted in **bold font**.

Architecture	AR@10↑	AR@100↑	AR@1000↑
$128-128-128-1$	0.097	0.218	0.317
$256-256-1$	0.098	0.218	0.316
$\mathbf{256-256-256-1}$	**0.100**	**0.221**	**0.320**
$256-256-256-256-1$	0.100	0.219	0.317
$512-512-512-1$	0.098	0.216	0.313

Finally, the superpixel classifier assigns each superpixel to the object proposal or the background based on the 257D feature vector (top right in Fig. 5.5). This classification is conducted individually for each superpixel. Hence, no interaction between superpixels takes place at this stage. The classifier itself, visualized in Fig. 5.6, is a simple network with four fully connected layers. This design is efficient and yields better results compared to architectures with more or fewer kernels and layers (see Tab. 5.4). Note that some superpixels may not be included in the superpixel segmentation window. These superpixels are directly assigned to the background for computational reasons.

Overall, the superpixel-based refinement module in SAM leads to refined object proposals based on an innovative combination of coarse initial AttentionMask proposals, superpixel segmentations, and learned features.

5.2.3 Post-processing of Refined Proposals

We further improve the refined proposals with a four-stage post-processing. The post-processing unifies the results and removes superpixel segmentation artifacts as well as duplicate proposals. As a first stage, we apply bilateral filtering [Tomasi and Manduchi, 1998] on the superpixel level similar to the superpixel-based guided filtering in Hu et al. [2017b] or the inter-superpixel voting in Chen et al. [2016b]. The bilateral filtering adapts the classification result of edge superpixels, i.e., superpixels with an ambiguous classification score[3]. All neighbors and second-order neighbors are gathered for an edge superpixel. Subsequently, bilateral filtering is applied to the classification scores based on the mean RGB color value per superpixel. Hence, adjacent superpixels with similar colors will have unified classification scores and capture uniformly colored object parts entirely.

The second and third stages of the post-processing in SAM are morphological opening and closing to remove typical antenna-like artifacts of the proposals, visualized in Fig. 5.7. These artifacts either reach from the proposal into the background or vice-versa and are introduced

[3]The classification score, the superpixel classifier's output, is ambiguous if it is neither close to 0 nor close to 1. We use a lower limit of 0.25 and an upper limit of 0.4 to describe ambiguous superpixels. Both values were determined empirically.

(a) Input image **(b)** W/o opening and closing **(c)** With opening and closing

Figure 5.7: Comparison of results produced by SAM without (b) and with (c) morphological post-processing based on opening and closing. Applying opening and closing removes the antenna-like artifacts highlighted by the arrows. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) is visualized per annotated object. Input image and annotations taken from LVIS dataset [Gupta et al., 2019].

by FH along single-pixel edges with uniform color. The opening with a 3×3 structuring element removes the antennas reaching out from the proposal. In contrast, the closing with the same structuring element fills the antennas reaching out from the background into the proposal. Finally, *Non-Maximum Suppression* (NMS) with IoU threshold 0.95 is applied to the results to remove near-duplicates. Near-duplicates are more likely to occur when constructing object proposals based on superpixels since proposals that vary by a few pixels will be mapped to the same combination of superpixels in SAM.

Overall, the proposed post-processing improves the quantitative and qualitative results of SAM as we will discuss in Sec. 5.4.3.

5.3 Training

After discussing the architecture of SAM, we present the end-to-end training strategy. Since SAM is based on AttentionMask, we will only discuss the changes in training compared to AttentionMask and refer the reader to Sec. 4.3 for the remaining parts of the training procedure. The first novelty in training SAM is the optimization of the superpixel segmentations, which is conducted outside the network (see Sec. 5.3.1). Based on the superpixel segmentations, the training targets for the SAM's superpixel classifier are generated from the pixel-precise annotations as outlined in Sec. 5.3.2. Finally, the loss function of SAM and the overall training strategy are presented in Sec. 5.3.3.

5.3.1 Superpixel Segmentation Optimization

We utilize FH [Felzenszwalb and Huttenlocher, 2004] to generate a superpixel segmentation for each level of the feature pyramid in SAM. FH originally has one parameter, k (see Eq. 3.11 in

Sec. 3.1.2), which controls the maximum difference between components for merging. To generate the best possible superpixel segmentations, we optimize k for each number of superpixels from Tab. 5.1 on the LVIS validation set with precisely annotated objects [Gupta et al., 2019]. While optimizing k, we only consider superpixel segmentations that roughly match ($\pm 10\%$) the desired number of superpixels across the validation set.

For the optimization, we transform the object annotations of the LVIS validation dataset to create a segmentation dataset. Hence, for every annotated object in an image, we create a binary segmentation as ground truth for this image. Additionally, we assign each of these ground truth segmentations to one or more pyramid levels in SAM according to the size constraints from AttentionMask's training strategy. This assignment leads to different ground truth segmentations per pyramid level. Since most images do not feature objects matching every pyramid level, we remove these images from the set of images for the respective pyramid levels. In contrast, some images include multiple objects matching the same pyramid level, leading to multiple ground truth segmentations similar to the BSD dataset [Martin et al., 2001].

After creating the ground truth segmentations, we optimize k and the superpixels' minimum size[4] based on the Overall Segmentation Quality (OSQ) following Stutz et al. [2018]. Overall, eight sets of parameters are generated for the eight pyramid levels in SAM. Note that we conducted the same optimization for the other superpixel segmentation methods utilized in our ablation studies in Sec. 5.4.3.

5.3.2 Superpixel-precise Ground Truth

In AttentionMask, the targets for the segmentation module are the pixel-precise object annotations from the dataset. This is different for the superpixel classifier in SAM. Since the classifier assigns entire superpixels to an object proposal or the background, the pixel-precise annotations have to be transformed to the level of superpixels. This process is non-trivial, as the superpixel segmentations do not capture all annotated object boundaries. Hence, minor undersegmentation errors, superpixels covering object and background, exist in the segmentations.

To approximate a pixel-precise annotation with the superpixels of a given superpixel segmentation, we propose an iterative scheme inspired by Chen et al. [2015b]. We start by calculating the overlap of each superpixel with the pixel-precise annotation. All superpixels that do not overlap with the annotated object are immediately assigned to the background (negative samples). Subsequently, we assign the superpixel that has the highest overlap with the pixel-precise annotation to the superpixel-precise annotation (positive samples). The remaining superpixels are processed in descending order of their overlap. If a superpixel increases the IoU between the superpixel-precise annotation and the pixel-precise annotation, we add it to the superpixel-precise annotation. This greedy processing leads to a superpixel-precise approximation (positive samples) of the pixel-precise annotation given a superpixel segmentation. As discussed above, the approximation does not perfectly match the pixel-precise annotation but exhibits a high quality as visible from the examples in Fig. 5.8.

[4]The superpixels' minimum size is an additional parameter of the FH implementation used by Stutz et al. [2018].

(a) Pixel-precise annotations **(b)** Superpixel-precise annotations

Figure 5.8: Original pixel-precise annotations (a) and our approximated superpixel-precise annotations (b) for two objects annotated in the LVIS dataset [Gupta et al., 2019]. We use FH [Felzenszwalb and Huttenlocher, 2004] to generate the superpixel segmentations. Note that the superpixel-precise annotations do not perfectly match the pixel-precise annotations due to undersegmentation errors in the superpixel segmentations. Base images and annotations taken from the LVIS dataset [Gupta et al., 2019].

Overall, the proposed transformation leads to the ground truth for the superpixel classifier in SAM. Note that the superpixel-precise ground truth is different for each pyramid level in SAM and changes for each superpixel segmentation method.

5.3.3 Loss Function and Training Strategy

Since SAM is based on AttentionMask, most of the overall training procedure stays unchanged. Only, the feature extractor and the superpixel classifier are added as learned components since the superpixel segmentations are generated outside the network. The superpixel classifier is directly trained using the superpixel-precise annotations described previously and the binary cross-entropy loss as the loss function. In contrast, the feature extraction is indirectly learned based on the superpixel classifier. Adding the new loss for the superpixel classifier $\mathcal{L}_{\text{spx}}$ leads to SAM's overall loss

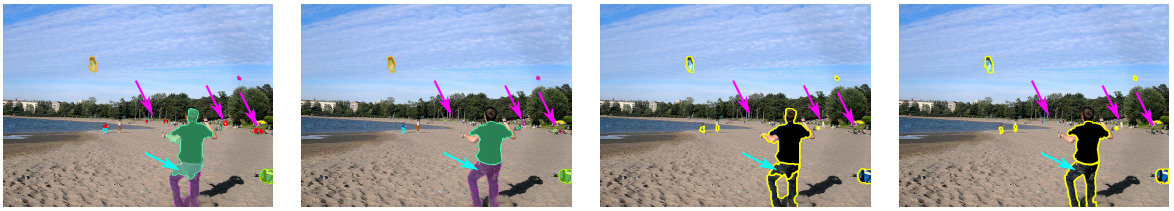
$$\mathcal{L} = w_{\text{spx}}\mathcal{L}_{\text{spx}} + w_{\text{objn}}\mathcal{L}_{\text{objn}} + w_{\text{att}}\mathcal{L}_{\text{att}} + w_{\text{seg}}\mathcal{L}_{\text{seg}} + w_{\text{SOAM}}\sum_i \mathcal{L}_{\text{SOAM}_i}, \tag{5.1}$$

based on AttentionMask's loss in Eq. 4.6. The weight w_{spx} balances the influence of the superpixel classifier loss similar to the other weights and is set to 1 in our experiments. The other weights remain unchanged compared to Sec. 4.3.2.

To initialize the ResNet-34-based backbone, we use ImageNet weights [He et al., 2016a]. The feature extractor, the superpixel classifier, and the remaining components are initialized from scratch. Similar to AttentionMask, we multiply the learning rate for the layers learned from scratch with a factor of 10. Overall, we train SAM end-to-end for 15 epochs with a learning rate of 0.0001, a momentum of 0.9, a weight decay of 0.00005, and batch size 1. As training data, we use the COCO training dataset [Lin et al., 2014] despite the imprecise annotations to allow a fair comparison to other systems.

5.4 Experiments

In this section, we compare SAM to state-of-the-art approaches in object proposal generation. The evaluation follows the setup in Sec. 4.4 and most object proposal generation literature [Pinheiro et al., 2015, 2016; Hu et al., 2017a]. Hence, we train each system on the COCO dataset [Lin et al., 2014] and assess the quality of the proposals in terms of Average



(a) Best proposal per annotated object **(b)** Annotated objects **(c)** Segmentation from proposals **(d)** Segmentation from annotated objects

Figure 5.9: Example for generating segmentations from the object proposals (a) and the annotated objects (b). The best matching proposals per annotated object (IoU ≥ 0.5) are joined to form a segmentation (c). Similarly, the annotations of discovered objects are joined to comprise the ground truth segmentation (d). Note that overlapping proposals lead to more than two regions as highlighted by the turquoise arrow in (c). Missed objects (IoU < 0.5) are omitted from both segmentations (see pink arrows). Base images and annotations taken from the LVIS dataset [Gupta et al., 2019].

Recall (AR, see Sec. 2.2.3) for different numbers of proposals and object sizes. Unlike existing evaluations, we evaluate not only on the challenging COCO test set but also on the complex LVIS test set [Gupta et al., 2019]. Evaluating on the LVIS dataset is important since it contains more precise annotations as discussed in Sec. 2.2.2. The precise annotations allow us to properly assess the adherence of the proposals to the boundaries of annotated objects. Nevertheless, we use the standard COCO dataset for training to allow a fair comparison. Note that we only evaluate pixel-precise proposals and omit box proposals, since we focus on the precise segmentation of objects in this chapter.

Besides the AR-based evaluation, we evaluate the generated proposals using Boundary Recall (BR) and Undersegmentation Error (UE) known from the evaluation of superpixel segmentations (see Sec. 2.1.3). To evaluate object proposals using BR and UE, we create a segmentation and a ground truth based on a system's object proposals. First, we select the best fitting proposal per discovered, annotated object (IoU ≥ 0.5). Subsequently, the binary segmentations of the selected proposals are joined across an image to generate the segmentation. Similarly, we join the annotated segmentation masks of the discovered objects, leading to the ground truth segmentation. Figure 5.9 depicts an example of this process, including the joint segmentation for the proposals (see Fig. 5.9(c)) and the annotated objects (see Fig. 5.9(d)). Finally, we evaluate the two joined segmentations using BR and UE to assess the adherence of the proposals to the annotated objects independent of the number of discovered objects. This is different from AR, which jointly measures how many objects are discovered and how well they are segmented.

Since we use FH superpixels in SAM, we denote our system as *SAM+FH* in the evaluation. We compare SAM+FH to AttentionMask (AttentionMask$_8^{128}$) from Ch. 4 and a variation of AttentionMask using the same ResNet-34 backbone as in SAM. Additionally, we compare to MCG [Arbeláez et al., 2014; Pont-Tuset et al., 2017] and COB [Maninis et al., 2016, 2017], which do not suffer from the inherent downsampling process within CNNs during object proposal generation. Finally, we compare SAM+FH to the CNN-based systems DeepMask [Pinheiro et al., 2015], SharpMask [Pinheiro et al., 2016], and FastMask [Hu et al., 2017a]. We do not include InstanceFCN [Dai et al., 2016] in the evaluation, since neither code nor results on the LVIS dataset are publicly available.

In the following, we first discuss the quantitative and qualitative results on the LVIS dataset featuring the precise annotations (see Sec. 5.4.1). Subsequently, Sec. 5.4.2 presents the

Table 5.5: Results on the LVIS test dataset using pixel-precise segmentation mask proposals in terms of six Average Recall (AR) measures. AR^S, AR^M, and AR^L denote results on small, medium, and large objects. See Tab. 2.1 for details on the AR variations. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Method	Backbone	AR@10↑	AR@100↑	AR@1k↑	AR^S@100↑	AR^M@100↑	AR^L@100↑
MCG	-	0.048	0.131	0.237	0.031	0.204	0.462
COB	-	0.054	0.148	0.281	0.043	0.235	**0.477**
DeepMask	ResNet-50	0.069	0.147	0.214	0.014	0.314	0.430
SharpMask	ResNet-50	0.073	0.154	0.229	0.014	*0.327*	0.460
FastMask	ResNet-50	0.069	0.161	0.256	0.055	0.296	0.386
AttentionMask	ResNet-50	0.073	*0.189*	*0.284*	0.081	0.312	0.446
AttentionMask	ResNet-34	*0.076*	0.185	0.271	*0.083*	0.305	0.423
SAM+FH	ResNet-34	**0.092**	**0.206**	**0.290**	**0.094**	**0.335**	*0.471*

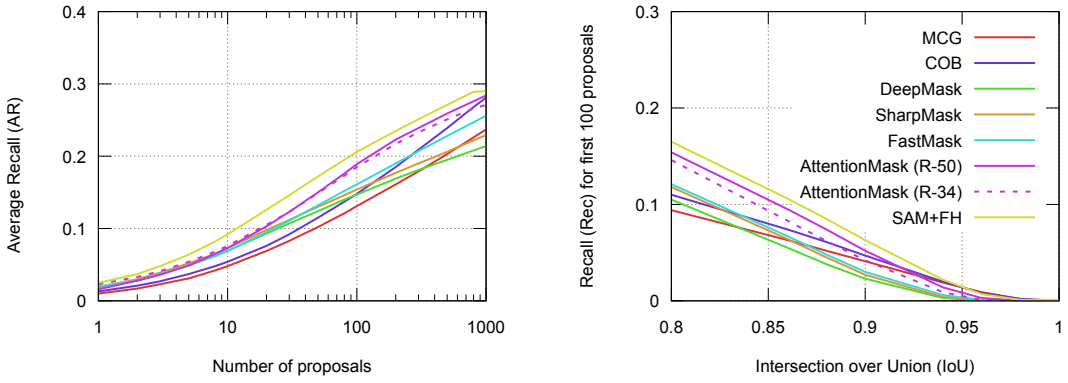
quantitative results on the COCO dataset with less precise annotations. Finally, three ablation studies in Sec. 5.4.3 show the influence of the superpixel segmentation method, the superpixel segmentation style, and the post-processing on the results of SAM+FH.

5.4.1 Results on the LVIS Dataset

General Object Proposal Generation Results

First, we present the general object proposal generation results in terms of AR on the complex LVIS test set. The results in Tab. 5.5 indicate that SAM+FH outperforms all other object proposal generation systems in terms of AR across all object sizes (AR@10, AR@100, and AR@1000). This is also visible from the plot in Fig. 5.10(a) depicting the AR across various numbers of proposals. Compared to AttentionMask using the ResNet-50 backbone, the AR@10 improves by 26.0%, while the AR@1000 still increases by 2.1%. Hence, SAM+FH compensates for the smaller backbone that leads to a drop in performance and even improves the results. Compared to AttentionMask using the smaller ResNet-34 backbone, the improvement of SAM+FH is even more significant (+7.0% in terms of AR@1000). This highlights the advantage of the superpixel-based refinement in SAM+FH since all other major components are identical to this AttentionMask variation. Note that the improvements in terms of AR over both AttentionMask variations are evident across all individual object sizes (+6.6% to +16.0%) as AR^S@100, AR^M@100, and AR^L@100 indicate.

SAM+FH also surpasses all other object proposal generation systems in terms of most AR measures. While outperforming FastMask by 13.3% in terms of AR@1000, SAM+FH surpasses SharpMask and DeepMask by 26.6% and 35.5%. The results of the systems w.r.t. to the object sizes differ from the results on the COCO dataset discussed in Sec. 4.4.1. For instance, DeepMask and SharpMask outperform FastMask on medium and large objects (AR^M@100 and AR^L@100). This behavior was not visible from the results on the COCO dataset due to the imprecise annotations. Since DeepMask and SharpMask follow the two-shot concept with less downsampling, the proposals are more precise compared to the one-shot systems like FastMask and AttentionMask. SAM+FH still outperforms DeepMask and SharpMask on large objects (+9.5% and +2.4%). The results in Tab. 5.5 also show that the proposals generated by MCG and COB without utilizing CNNs during object proposal generation are competitive on the LVIS dataset. For instance, COB outperforms DeepMask, SharpMask, and FastMask



(a) **(b)**

Figure 5.10: Detailed quantitative results on the LVIS test set. (a): Average Recall (AR) across various numbers of object proposals for the eight object proposal generation systems evaluated in this section. (b): Recall (Rec) of the eight object proposal generation systems for objects discovered with an IoU of at least 0.8 (100 proposals). R-50 and R-34 denote variations of AttentionMask based on ResNet-50 and ResNet-34 backbones. Note that the y-axes are truncated at 0.4 (a) and 0.3 (b) for improved visibility.

on AR@1000. This is due to the precise segmentations using superpixels. However, the lack of a high-quality ranking prevents MCG and COB from outperforming those systems on less than 500 proposals (see Fig. 5.10(a)). Still, COB outperforms even SAM+FH on large objects since it does not suffer from the inherited downsampling in CNNs.

Quality of the Segmentation Masks

After discussing the general object proposal generation results in terms of AR, we focus on the quality of the proposals' segmentation masks. Figure 5.10(b) gives a more detailed overview of the Recall (Rec) for the first 100 proposals across different IoU levels above 0.8. Hence, all IoU levels in Fig. 5.10(b) lead to proposals that precisely capture the objects. Up to an IoU of 0.95 (Rec(0.95)), SAM+FH constantly outperforms all other systems. Especially on more precise IoU levels, the relative improvement of SAM+FH is substantial. For instance, SAM+FH outperforms AttentionMask by 21.1% at an IoU of 0.9 (Rec(0.9)). Above an IoU of 0.95, MCG and COB surpass all CNN-based approaches, including SAM+FH. However, Rec is generally low in this range (up to 0.009).

To evaluate the quality of the segmentation masks in more detail, we use BR and UE to assess the segmentation quality independent of the number of discovered objects as discussed above. Table 5.6 presents the results of this analysis across all object sizes and for each object size (S, M, and L) individually. The results show that SAM+FH outperforms all other CNN-based systems in terms of BR and UE. Hence, object proposals generated by SAM+FH adhere better to the boundaries of the annotated objects based on utilizing superpixels. While the improvement for small objects is low (+2.9% in terms of BR compared to AttentionMask), proposals for medium and large objects strongly benefit from the refinement. For instance, SAM+FH outperforms AttentionMask by 12.7% and 43.2% in terms of BR on medium and large objects. This is due to the stronger effects of the downsampling process on medium and large objects in CNNs. Similar to Fig. 5.10(b), the results show that MCG and COB generate more precise object proposals than the CNN-based systems including SAM+FH. However,

Table 5.6: Detailed results on the LVIS test dataset using pixel-precise segmentation mask proposals in terms of Boundary Recall (BR) and Undersegmentation Error (UE). BR^S/UE^S, BR^M/UE^M, and BR^L/UE^L denote results on small, medium, and large objects. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Method	Backbone	BR↑	UE↓	BR^S↑	UE^S↓	BR^M↑	UE^M↓	BR^L↑	UE^L↓
MCG	-	*0.685*	0.073	0.833	**0.004**	0.709	0.023	*0.614*	*0.089*
COB	-	**0.734**	**0.059**	0.823	0.005	**0.738**	0.021	**0.686**	**0.068**
DeepMask	ResNet-50	0.488	0.087	0.727	0.006	0.622	0.024	0.308	0.109
SharpMask	ResNet-50	0.561	0.080	0.782	0.005	0.681	0.023	0.383	0.100
FastMask	ResNet-50	0.510	0.084	0.794	0.006	0.622	0.023	0.318	0.107
AttentionMask	ResNet-50	0.568	0.070	*0.840*	0.005	0.644	*0.020*	0.389	0.091
AttentionMask	ResNet-34	0.547	0.075	0.832	0.005	0.637	*0.020*	0.356	0.099
SAM+FH	ResNet-34	0.681	*0.068*	**0.864**	**0.004**	*0.726*	**0.019**	0.557	0.090

as the AR-based results indicate (see Tab. 5.5), MCG and COB discover fewer objects than SAM+FH.

Qualitative Results

The qualitative results in Fig. 5.11 support the general findings of the quantitative results. Across all results in Fig. 5.11, SAM+FH proposals adhere better to the annotated objects and capture fine details or complete the object shapes. For instance, the first two rows show that neither FastMask nor AttentionMask is able to precisely capture details like the animals' snout, ears, or individual legs. SAM+FH captures such details utilizing superpixels. This is highlighted by precise segmentations of the ears in both examples and the tiny legs in the second example. Similarly, the results in the rows three and four also show an improved adherence of the proposals generated by SAM+FH to the annotated objects. Even small details of the fire hydrant or the tennis player's pants are mostly captured precisely.

Rows five and six show typical examples of SAM+FH completing object shapes. FastMask and AttentionMask only partially capture the airplane as well as the kite due to missing small details like the tail of the kite. SAM+FH completes the initial coarse proposals and captures most details using precise superpixels. The final two rows depict examples of very challenging scenes. The scene in the seventh row is highly complex with partially overlapping objects. Nevertheless, SAM+FH refines the objects and precisely captures most object corners using superpixels. In contrast, the proposals generated by FastMask and AttentionMask suffer from smoothed object corners, despite a high contrast and simple shapes. The smooth shapes are due to the interpolation of the coarse proposals. The last row shows an example of small objects. While the proposals of SAM+FH capture the tiny snouts, tails, and most fins of the seahorses, FastMask and AttentionMask propose blob-like proposals that miss several details. These findings are in line with the results in Tab. 5.5 and Tab. 5.6 on small objects.

Despite the generally high quality of the proposals generated by SAM+FH, few typical errors remain. For instance, undersegmentation errors in the superpixel segmentations pose a problem for SAM+FH. Such errors lead to proposals that partially cover both object and background or multiple objects as visible for the monitors in the seventh example in Fig. 5.11. Due to FH's subpar segmentation quality in terms of UE, several proposals across the test set

Figure 5.11: Qualitative results of FastMask [Hu et al., 2017a], AttentionMask based on a ResNet-50 (see Ch. 4), and SAM+FH on images of the LVIS test dataset. The arrows highlight prominent differences between the systems. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) is visualized per annotated object. Input images and annotations taken from the LVIS dataset [Gupta et al., 2019].

Table 5.7: Results on the COCO test dataset using pixel-precise segmentation mask proposals in terms of six Average Recall (AR) measures. AR^S, AR^M, and AR^L denote results on small, medium, and large objects. See Tab. 2.1 for details on the AR variations. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Method	Backbone	AR@10↑	AR@100↑	AR@1k↑	AR^S@100↑	AR^M@100↑	AR^L@100↑
MCG	-	0.077	0.186	0.299	0.041	0.182	0.435
COB	-	0.104	0.233	0.383	0.065	0.243	0.501
DeepMaskZoom	ResNet-50	0.151	0.286	0.371	0.093	0.389	0.466
SharpMask	ResNet-50	0.154	0.278	0.360	0.035	0.399	0.513
SharpMaskZoom	ResNet-50	0.156	0.304	0.401	0.099	*0.412*	0.495
InstanceFCN	ResNet-50	0.166	0.317	0.392	-	-	-
FastMask	ResNet-50	0.169	0.313	0.406	0.106	0.406	0.517
AttentionMask	ResNet-50	*0.180*	**0.349**	**0.444**	*0.162*	**0.421**	*0.560*
AttentionMask	ResNet-34	0.174	0.332	0.420	0.154	0.401	0.534
SAM+FH	ResNet-34	**0.193**	*0.346*	*0.421*	**0.164**	0.411	**0.561**

are affected by such errors. Another typical source of errors are misclassified superpixels as visible around the pants of the tennis player in the fourth row. A superpixel adjacent to the crotch of the pants is misclassified as part of the proposal. Similarly, one hoof of the larger cow in the first example and the tip of the fire hydrant in the third example are missed by SAM+FH due to misclassified superpixels.

Overall, the qualitative results showcase that the proposals generated by SAM+FH adhere better to the object boundaries compared to FastMask and AttentionMask despite few errors. This supports the findings of the quantitative evaluations based on AR, BR, and UE. The remaining errors are related to undersegmentation errors in the FH superpixel segmentations or misclassifications of individual superpixels.

5.4.2 Results on the COCO Dataset

For completeness, we briefly discuss the results of SAM+FH on the COCO test set, which features imprecise annotations. Table 5.7 presents the results of this evaluation, which is similar to Sec. 4.4.1. As expected, SAM+FH does not generally outperform AttentionMask, since the imprecise annotations dismiss the more precise object proposals created by SAM+FH. As a result, only in terms of AR@10 SAM+FH outperforms AttentionMask across all object sizes, which is attributed to the superpixel-driven NMS. Additionally, the results are almost identical when comparing AttentionMask with the ResNet-34 backbone and SAM+FH in terms of AR@1000. Hence, without NMS, both systems perform almost at an identical level. However, the overall results show that SAM+FH outperforms most other object proposal generation systems since it is based on AttentionMask.

5.4.3 Ablation Studies

After discussing the overall results for SAM+FH, we present three ablations studies that analyze the influence of the superpixel segmentation method, the style of the superpixel segmentations, and the proposed post-processing. Additional ablation studies regarding design decisions were already presented in Sec. 5.2 and Sec. 5.3.

Table 5.8: Results of SAM+FH using different numbers of superpixels and FH superpixel segmentations enhanced with all boundaries from the annotated objects (with GT). We use SAM+FH with only five pyramid levels and generate the results on the LVIS validation set. AR denotes the Average Recall for the first 10, 100, or 1000 proposals. **Bold font** highlights the best result without the boundaries from the annotated objects, while *italic font* indicates the best results utilizing the boundaries from the annotated objects.

Superpixel segmentation	Number of superpixels	AR@10↑	AR@100↑	AR@1000↑
FH	$8000 - 500$	**0.100**	**0.221**	**0.320**
FH	$4000 - 250$	0.098	0.213	0.310
FH with GT	$8000 - 500$	0.108	0.235	0.340
FH with GT	$4000 - 250$	*0.117*	*0.259*	*0.361*

Superpixel Segmentation Methods

The choice of a proper superpixel segmentation is critical when applying SAM+FH. As described in Sec. 5.2.1, we use 8000 to 500 FH superpixels for the different levels of the feature pyramid in SAM. First, we evaluate the number of superpixels per pyramid level. The upper part of Tab. 5.8 shows the results using 8000 - 500 FH superpixels and 4000 - 250 FH superpixels. As expected, more superpixels lead to improved performance. However, the difference is small, with an improvement of at most 3.8%. Utilizing even more superpixels, e.g., 16000 - 1000, is impossible due to the limited memory on the GPU[5]. In a related experiment, we add all boundaries from the annotated objects to the superpixel segmentations and evaluate our system again. The results of this experiment in the lower part of Tab. 5.8 indicate that the presence of all annotated object boundaries leads to better results with less superpixels. This implies that a strong oversegmentation is not beneficial for SAM.

Moving to different superpixel segmentation methods, we evaluate the influence of seven methods on SAM: FH [Felzenszwalb and Huttenlocher, 2004], ETPS [Yao et al., 2015], WS [Meyer, 1994], ERS [Liu et al., 2011], SLIC [Achanta et al., 2012], NC [Shi and Malik, 2000], and SEEDS [Van den Bergh et al., 2015]. The results in Tab. 5.9 show that FH leads to the best results ahead of ETPS. This is in contrast to the findings of Stutz et al. [2018] on general superpixel segmentation. In Stutz et al. [2018], FH is ranked in the lower third of 28 evaluated methods. Specifically, FH is ranked behind all other methods evaluated in this experiment. Moreover, SEEDS, which is highly ranked in Stutz et al. [2018], produces subpar results when applied in SAM. These results are related to the different segmentations styles and the amount of oversegmentation as we will examine in the subsequent ablation study.

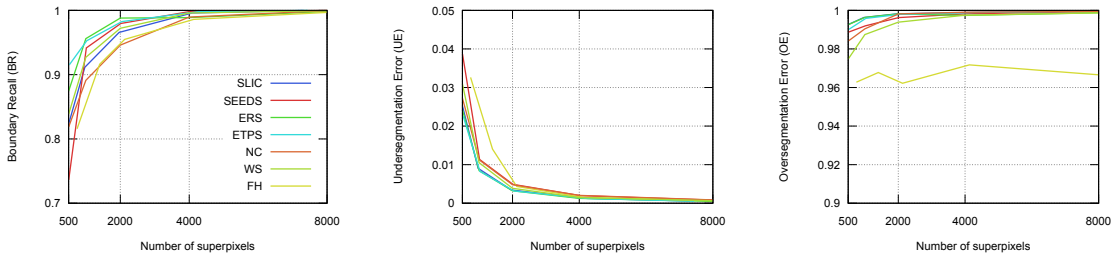
Superpixel Segmentation Quality

To further investigate the preferred superpixel segmentation style of SAM, we evaluate the superpixel segmentation methods utilized in the previous ablation study in terms of BR, UE, and Oversegmentation Error (OE) on the LVIS validation set. As ground truth, we use the joined binary segmentations of the annotated objects. The evaluation in terms of BR in

[5]We assume a 12 GB GPU here.

Table 5.9: Results of SAM using different superpixel segmentation methods. The superpixel segmentation methods are FH [Felzenszwalb and Huttenlocher, 2004], ETPS [Yao et al., 2015], WS [Meyer, 1994], ERS [Liu et al., 2011], SLIC [Achanta et al., 2012], NC [Shi and Malik, 2000], and SEEDS [Van den Bergh et al., 2015]. We use SAM with only five pyramid levels and generate the results on the LVIS validation set. AR denotes the Average Recall for the first 10, 100, or 1000 proposals. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Superpixel segmentation	Number of superpixels	AR@10↑	AR@100↑	AR@1000↑
FH	8000 – 500	**0.100**	**0.221**	**0.320**
ETPS	8000 – 500	*0.097*	*0.214*	*0.311*
WS	8000 – 500	0.096	0.212	0.303
ERS	8000 – 500	0.090	0.208	0.309
SLIC	8000 – 500	0.089	0.200	0.296
NC	8000 – 500	0.093	0.197	0.272
SEEDS	8000 – 500	0.082	0.182	0.266



(a) Boundary Recall **(b)** Undersegmentation Error **(c)** Oversegmentation Error

Figure 5.12: Quantitative results of seven superpixel segmentation methods on the LVIS validation set in terms of Boundary Recall (BR) (a), Undersegmentation Error (UE) (b), and Oversegmentation Error (OE) (c) across 500 to 8000 superpixels. The superpixel segmentation methods are FH [Felzenszwalb and Huttenlocher, 2004], ETPS [Yao et al., 2015], WS [Meyer, 1994], ERS [Liu et al., 2011], SLIC [Achanta et al., 2012], NC [Shi and Malik, 2000], and SEEDS [Van den Bergh et al., 2015].

Fig. 5.12(a) and UE in Fig. 5.12(b) reveals similar results as in Stutz et al. [2018] since ETPS and ERS outperform most other methods. However, the evaluation in terms of OE shows that WS and mainly FH outperform the other methods, although the results are generally on a high level[6]. The results for FH and WS are in line with the fact that both methods produce less oversegmentation than other methods by design. Linking these results with the results of SAM utilizing different superpixel segmentations (see Tab. 5.9) indicates that SAM prefers superpixel segmentations with a lower OE. These findings are supported by the results of SAM utilizing superpixel segmentations augmented with the boundaries of the annotated objects. Those results (see Tab. 5.8) also revealed that less oversegmentation is helpful for improved performance. Overall, a low OE with a good segmentation quality in terms of BR and UE are essential for strong results of SAM.

The effects of a low OE are also visible in Fig. 5.13 that presents superpixel segmentations of three LVIS images generated with FH, ETPS, and SLIC. As expected, FH leads to

[6]The generally high level of OE is due to the definition of OE (see Sec. 2.1.3) and the existence of a dominant background region in the created ground truth.

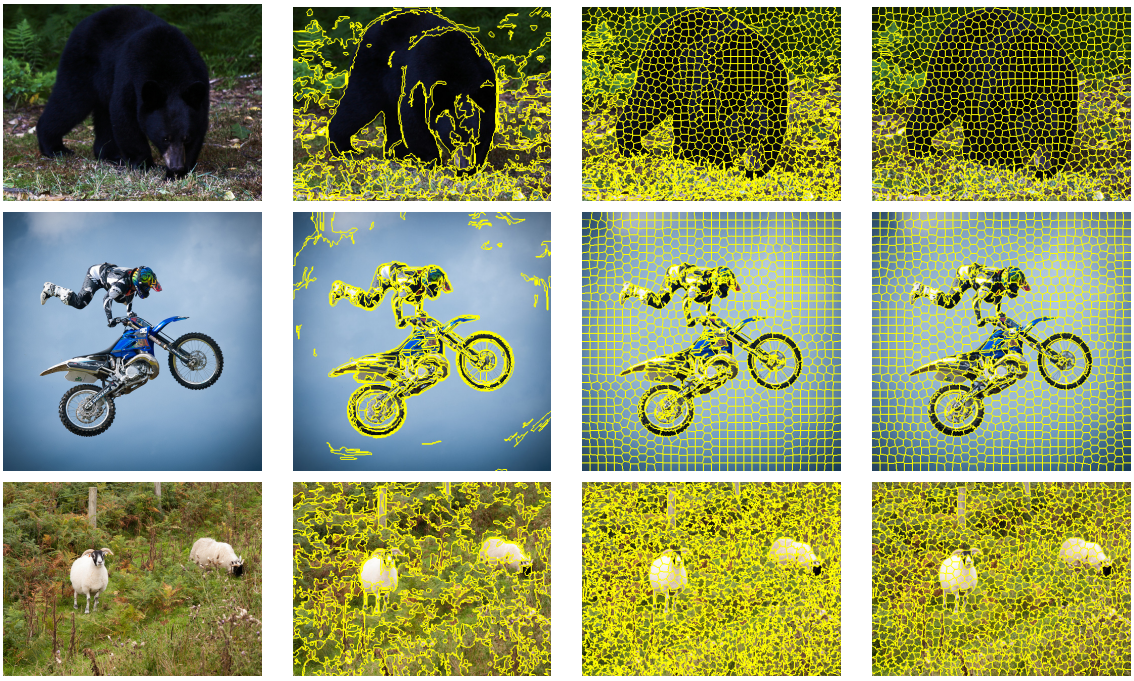


Figure 5.13: Qualitative superpixel segmentation results produced by FH [Felzenszwalb and Huttenlocher, 2004], ETPS [Yao et al., 2015], and SLIC [Achanta et al., 2012] for three images from the LVIS validation set [Gupta et al., 2019].

superpixels that are non-uniformly distributed across the images with large superpixels covering homogeneous foreground (first row) or background (second row) areas. Thus, the FH superpixel segmentations exhibit less oversegmentation. Hence, major parts of the image are easily classified as object or background. The third row shows an example of a highly textured image. Nevertheless, the FH superpixel segmentation shows the desired behavior of a limited amount of oversegmentation (see white sheep) compared to ETPS or SLIC. Overall, the difference in oversegmentation between FH, ETPS, and SLIC, which leads to the different results in SAM, is well visible.

Influence of the Post-processing

As a final ablation study, we examine the effect of the four-stage post-processing in SAM. The first part of Tab. 5.10 presents the results of SAM+FH without the post-processing and the results after successively adding each post-processing stage. These results show that the post-processing improves the performance of SAM+FH by 21.1% in terms of AR@10. Additionally, as the intermediate results indicate, all four stages lead to improvements. Especially the bilateral filtering on superpixel-level and the near-duplicate removal lead to substantial improvements. While the bilateral filtering improves the results across all numbers of proposals, the near-duplicate removal only improves the AR for 10 and 100 proposals. Hence, removing potential near-duplicates might accidentally dismiss proposals that better fit an annotated object than the remaining proposals. Note that the morphological operations mainly have a positive effect on the qualitative results as the example in Fig. 5.7 revealed.

Table 5.10: Results of the proposed superpixel-based post-processing stages in SAM+FH (upper part). The results applying the post-processing on a pixel-level are given in the lower part of the table. All results are generated on the LVIS test set. AR denotes the Average Recall for the first 10, 100, or 1000 proposals. **Bold font** highlights the best results.

Post-processing	AR@10↑	AR@100↑	AR@1000↑
None	0.076	0.183	0.275
Bilateral filtering	0.079	0.190	0.288
+ Opening	0.079	0.191	0.292
+ Closing	0.079	0.192	**0.293**
+ Near-duplicate removal	**0.092**	**0.206**	0.290
All pixel-based	0.087	0.194	0.272

Since all post-processing stages are applicable on the level of pixels, we present the results of the four-stage post-processing on pixel-level in the lower part of Tab. 5.10. Although the pixel-level post-processing improves the initial results, the post-processing on the superpixel-level leads to a better performance. This is mainly due to the bilateral filtering that re-assigns entire superpixels.

5.5 Discussion

This chapter introduced our novel object proposal generation system SAM as an extension of AttentionMask. SAM addresses the problem of coarse object proposals generated by CNN-based systems with an innovative combination of superpixels and the coarse CNN-based proposals. Starting from the coarse AttentionMask proposals, SAM utilizes CNN-based features, superpixel segmentations, superpixel pooling, and a new superpixel classifier to create refined proposals. These proposals adhere better to the object boundaries while maintaining a high recall. Thus, SAM combines the advantages of CNN-based systems with the advantages of traditional superpixel-based object proposal generation systems in an innovative manner.

The evaluation on the challenging LVIS dataset showed that SAM with FH superpixels (SAM+FH) outperforms AttentionMask by up to 26.0% across all object sizes. An even more substantial improvement is visible compared to other modern object proposal generation systems. By assessing the segmentation quality of the proposals, we revealed that the proposals generated by SAM+FH adhere better to the object boundaries compared to all other CNN-based systems. Additionally, multiple ablation studies unveiled that the choice of the superpixel segmentation method in SAM is crucial. Based on multiple experiments, we showed that a superpixel segmentation with less oversegmentation like FH leads to the best overall results, despite a subpar segmentation quality in terms of BR and UE.

This leads to a major limitation of SAM+FH. Since FH's segmentation quality in terms of BR and UE is worse compared to other methods like ETPS and SLIC, several undersegmentation errors remain in the superpixel segmentations. These errors restrict the overall performance of SAM+FH, since it is impossible to capture object boundaries that are missed by the superpixels. Hence, improved superpixel segmentations combining low oversegmentation with a good segmentation quality are desirable. In addition to the imperfect superpixel segmentations,

some limitations inherited from AttentionMask, like the impaired discovery of tiny objects or the suboptimal ranking, remain. However, the ranking of object proposals slightly improved due to the near-duplicate removal based on superpixels.

Overall, we introduced our novel, innovative object proposal generation system SAM in this chapter that substantially improves the adherence of object proposals to the object boundaries. However, the performance of SAM is restricted by the quality of the FH superpixel segmentations, which we will address in Ch. 6 and Ch. 7 from different directions.

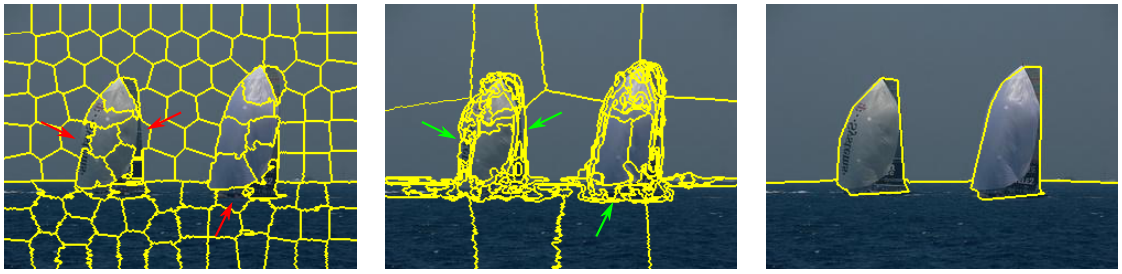
Chapter 6

Edge-adaptive Superpixel Segmentations

Table of Contents

The previous chapter introduced our innovative object proposal generation system SAM and showed that superpixels are helpful to refine coarse CNN-based object proposals. The evaluation of SAM revealed that superpixel segmentation methods producing a limited amount of oversegmentation, i.e., exhibiting a low Oversegmentation Error (OE), lead to best results in SAM. Therefore, modern superpixel segmentation methods that perform well on superpixel segmentation benchmarks [Stutz et al., 2018] like ETPS [Yao et al., 2015] or SLIC [Achanta et al., 2012] did not outperform older methods like FH [Felzenszwalb and Huttenlocher, 2004]. These findings align with the characteristics of ETPS and SLIC that ignore the different levels of detail in an image and uniformly distribute superpixels across the image. As a result, SLIC and ETPS lead to more oversegmentation than FH.

This segmentation style of SLIC and ETPS is rooted in the uniform initialization of the methods. For instance, SLIC starts from initial seeds placed on a regular grid across the image. Many superpixel seeds are placed in uniform areas like the sky in Fig. 6.1(a), while less uniform areas are covered with the same seed resolution. Hence, the initialization does not reflect the different levels of detail in an image. Although SLIC allows superpixels to shift their position to adapt to the image content, these shifts only occur if the pixels around a seed are non-uniform. As a result, uniform areas are heavily oversegmented (high OE), while details like the contours of the boats in Fig. 6.1 are missed, leading to undersegmentation errors (see red arrows). Overall, SLIC wastes superpixels in uniform areas. Hence, the same number of superpixels



(a) SLIC **(b)** EA-SLIC **(c)** Ground truth

Figure 6.1: Superpixel segmentation results of SLIC [Achanta et al., 2012] (a) and our proposed edge-adaptive framework using SLIC denoted as EA-SLIC (b) as well as the ground truth (c). While the SLIC superpixels are almost uniformly distributed across the image, our edge-adaptive framework allows SLIC to generate more superpixels in image areas with more details. As a result, fewer superpixels cover the background, while the boundaries of the boats are covered with more superpixels. This leads to less oversegmentation and a better segmentation quality compared to SLIC. The red arrows denote missed boundaries (undersegmentation errors), while the green arrows highlight the successful recognition of those boundaries. Input image and annotation taken from the SBD dataset [Gould et al., 2009].

would allow a better segmentation of details given a non-uniform superpixel distribution (see Fig. 6.1(b)). Similar problems exist in other superpixel segmentation methods [Yao et al., 2015; Van den Bergh et al., 2015; Lee et al., 2017].

Few strategies have been introduced to adapt the uniform distribution of superpixels or supervoxels in segmentation methods [Weikersdorfer et al., 2013; Kanezaki and Harada, 2015; Gao et al., 2017; Zhang et al., 2021]. The strategies adapt the uniform initialization to the levels of detail found in different image areas based on depth [Weikersdorfer et al., 2013], color [Kanezaki and Harada, 2015], or saliency [Gao et al., 2017; Zhang et al., 2021] information. Since we focus on segmenting objects, we propose a new edge-adaptive strategy. Our edge-adaptive superpixel segmentation framework uses edge detection results to estimate the level of detail for adapting the uniform distribution. Edge detection results fit our object proposal-centered workflow better since edges usually surround objects marking their physical embodiment. In contrast, objects may not be salient or colorful, while depth data is unavailable for most datasets in object proposal generation.

Utilizing the edge detection results in our novel edge-adaptive strategy, we follow Gao et al. [2017] and cluster the image pixels based on the density of strong edges. For each cluster, we adapt the superpixel resolution[1] to the level of detail based on the edge density. Hence, areas around objects that exhibit a higher edge density are covered with more superpixels, while uniform areas lead to fewer superpixels (see Fig. 6.1(b)). This edge-adaptive strategy allows arbitrary superpixel segmentation methods like SLIC or ETPS to reduce the oversegmentation while maintaining a strong segmentation quality. As a result, the reduced oversegmentation makes them better suited for application in SAM.

This chapter presents our novel edge-adaptive superpixel segmentation framework based on our publication Wilms and Frintrop [2017]. First, we briefly discuss related approaches for adapting segmentation methods in Sec. 6.1. Section 6.2 introduces our edge-adaptive framework for arbitrary superpixel segmentations based on the clustering of edge detection

[1]Superpixel resolution denotes the number of superpixels per unit of area. If the superpixel resolution varies across the image, the superpixel segmentation exhibits a non-uniform spatial distribution of superpixels.

results. Subsequently, we evaluate the edge-adaptive superpixel segmentation framework utilizing SLIC and ETPS on the superpixel segmentation task in Sec. 6.3. This is followed by an evaluation of SAM with the edge-adaptive superpixel segmentations in Sec. 6.4. We conclude this chapter with a discussion of the advantages and limitations of the proposed framework w.r.t. to both applications in Sec. 6.5.

6.1 Related Work on Adapting Segmentations

Few approaches were proposed to adapt the uniform distribution of segmentation methods based on depth [Weikersdorfer et al., 2013], color [Kanezaki and Harada, 2015], or saliency [Gao et al., 2017; Zhang et al., 2021] information. All four approaches adjust the uniform distribution of superpixels or supervoxels by adapting the uniform initialization of the segmentation methods. For instance, Weikersdorfer et al. [2013] use depth information of RGB-D data to adapt the initialization of a SLIC-like segmentation method. Areas far from the camera are segmented with more superpixels than closer areas. The adapted initialization is generated by applying a blue-noise sampling method to the depth information. This leads to a non-uniform distribution of initial seed locations for the SLIC-like segmentation of Weikersdorfer et al. [2013].

Utilizing color information, Kanezaki and Harada [2015] adapt the initial seeding of the SLIC-like supervoxel segmentation method VCCS (*Voxel Cloud Connectivity Segmentation*) [Papon et al., 2013] for point clouds. Originally, VCCS applies a uniform seeding similar to SLIC. However, since not every location in a scene is occupied by a voxel in a point cloud, the seeds are moved into the closest voxel of the point cloud. Finally, seed points that are too close to each other are merged. Kanezaki and Harada [2015] change this initial seeding by clustering the voxels based on their RGB values and applying the original seeding to each cluster individually. Since the seeds move to the closest voxel for each cluster, regions with multiple colors will have voxels in multiple clusters, leading to more seeds. Finally, the seeds of all clusters are merged to create the non-uniform overall seeding that leads to a non-uniform distribution of supervoxels.

Gao et al. [2017] and Zhang et al. [2021] utilize saliency to adapt the uniform initialization of different methods. Also tackling the segmentation of point clouds with VCCS, Gao et al. [2017] apply k-means clustering to the voxels based on saliency information. Subsequently, each cluster in the point cloud receives an individual seeding resolution. Highly salient clusters receive a denser seeding and vice versa. Focusing on RGB images, Zhang et al. [2021] adapt the seeding of a SLIC variation. The seed generation is based on an iterative method that selects the most salient point as seed and subsequently inhibits the seed and its neighborhood. This process is iteratively applied until the maximum number of seeds is reached. Like Gao et al. [2017], this leads to a denser seed resolution in highly salient areas and vice versa. Overall, both approaches lead to a non-uniform distribution of superpixels or supervoxels.

In general, three different types of prior information have been utilized to adapt the initialization of superpixel or supervoxel segmentation methods. However, neither of the three types properly estimates the level of detail for an object centered processing. In this context, detailed segmentations of object boundaries and crowded images regions are needed to improve subsequent results. While Weikersdorfer et al. [2013] densely segment all distant areas independent of the level of detail, the saliency-based approaches [Gao et al., 2017; Zhang et al.,

2021] concentrate the generation of superpixels on salient areas only. Utilizing color [Kanezaki and Harada, 2015] is closest to match the mentioned criteria. However, the approach also focuses on textures within objects and misses objects of similar color that are separated by an edge. In contrast to these methods, our edge-adaptive superpixel segmentation framework is more general and captures the level of detail by focusing on object boundaries and crowded image regions.

6.2 Edge-adaptive Superpixel Segmentation Framework

After reviewing related approaches, we present our novel edge-adaptive superpixel segmentation framework. Figure 6.2 visualizes the overall structure of the framework. First, as described in Sec. 6.2.1, we extract strong edges from the input image by applying the SE edge detector [Dollár and Zitnick, 2013, 2014] and a thresholding step. Subsequently, we generate an edge density that represents the number of strong edges around each pixel and guides the adaptation of the superpixel distribution in our framework. Image areas with high edge density are segmented with more superpixels and vice versa. To identify such image areas, we cluster the edge density using k-means. For each cluster, we apply a superpixel segmentation method with an adapted superpixel resolution as detailed in Sec. 6.2.2. Finally, we join the superpixel segmentations from the different clusters and adjust the overall number of superpixels if necessary (see Sec. 6.2.3). This joining combines the different superpixel resolutions into one final superpixel segmentation. To generate the best results, we optimize the framework's parameters as discussed in Sec. 6.2.4.

6.2.1 Edge Detection and Processing

Our proposed edge-adaptive superpixel segmentation framework starts by extracting edges from the input image. For the extraction of edges, we utilize the *Structured Edges* (SE) edge detector [Dollár and Zitnick, 2013, 2014]. SE generates edges using a structured random forest predicting edge maps for small tiles of the input image. Compared to other approaches like Canny [Canny, 1986], gPb [Arbeláez et al., 2010], HED [Xie and Tu, 2015], or COB [Maninis et al., 2016, 2017], SE edges yield better results in the edge-adaptive superpixel segmentation framework. This is visible from the results in Fig. 6.3 that show a slightly better performance of SE over other methods in terms of Overall Segmentation Quality (OSQ).

To remove weak edge pixels that are unlikely to belong to an edge, we binarize the edge detection result using a threshold τ. This binarization makes the subsequent edge density independent of the edge likelihood. Treating all non-weak edge pixels equally is reasonable since strong edges will be captured by a superpixel segmentation method anyway. Hence, more superpixels are not necessary in these areas. The result of the first two steps is the binarized edge map as visualized in the upper part of Fig. 6.2.

To generate an edge density at each pixel, we apply a Gaussian filter to the binarized edge map (top right in Fig. 6.2). Hence, the edge density is the weighted average of the surrounding edge pixels. To adjust the size of the neighborhood, we optimize the standard deviation σ of the Gaussian filter as outlined in Sec. 6.2.4. Additional edge density examples are visualized in the second column of Fig. 6.4. Those examples showcase that the edge density focuses on

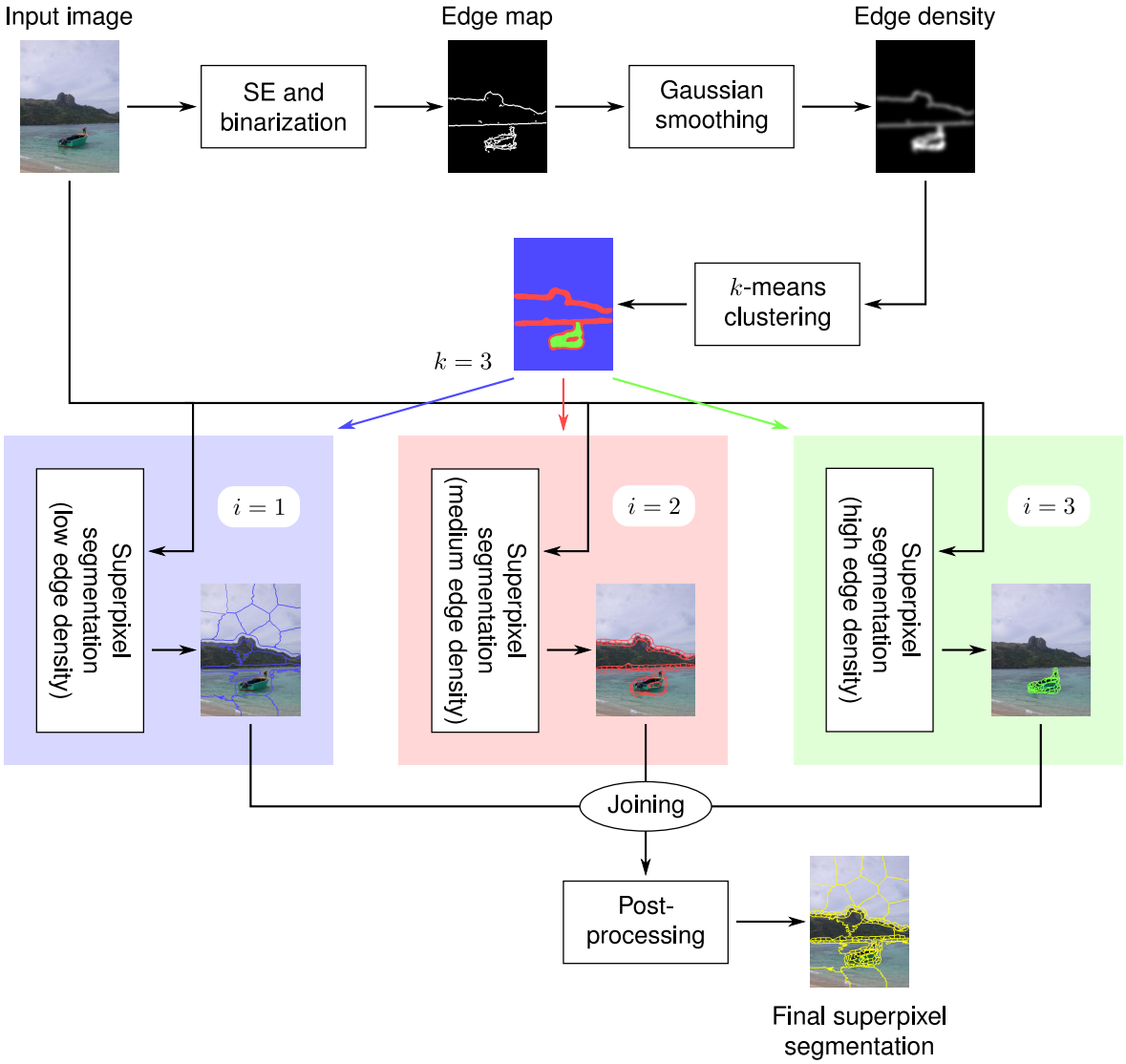
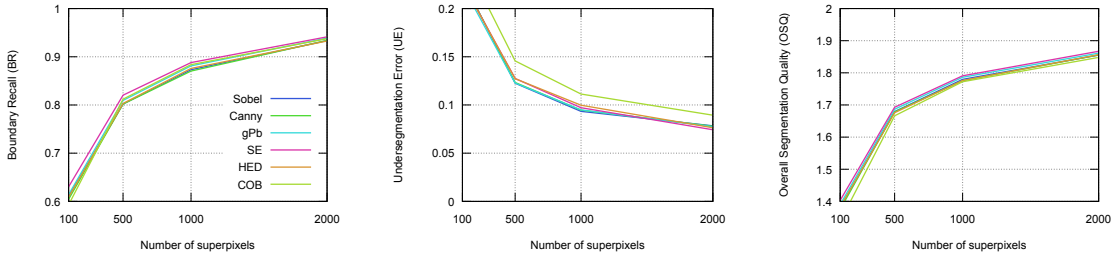
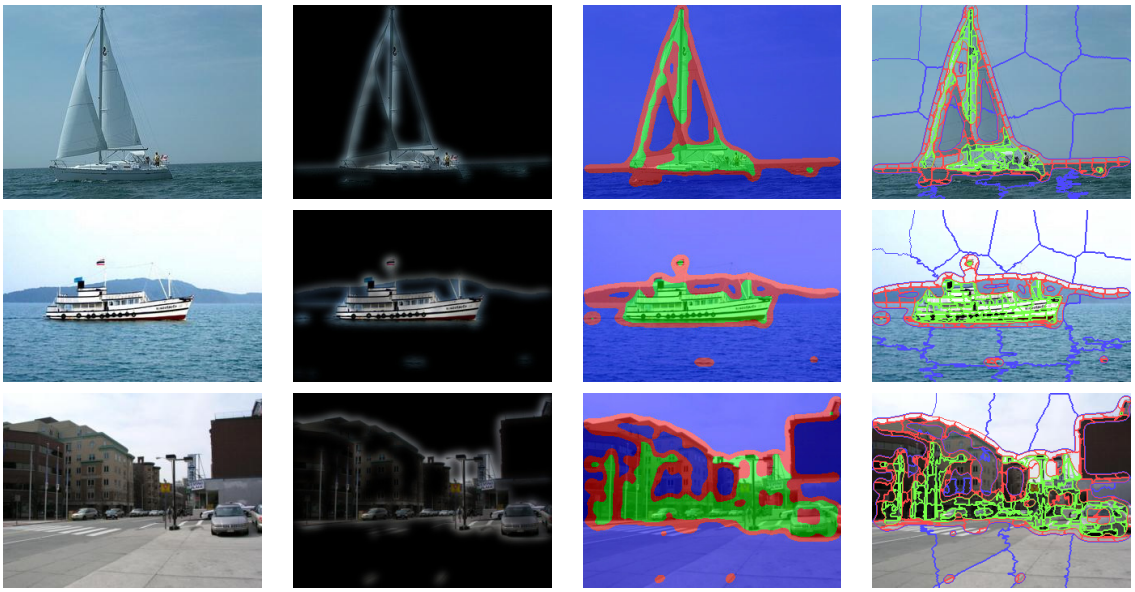


Figure 6.2: Overview of the proposed edge-adaptive superpixel segmentation framework. First, strong edges are extracted from the image using the Structured Edges (SE) [Dollár and Zitnick, 2013, 2014] edge detector and a binarization step. This leads to an edge map that is subsequently smoothed to generate the edge density. Second, based on the edge density values, the pixels are clustered with k-means clustering (here $k = 3$) to separate image areas of different edge densities (blue, red, and green). Finally, the input image is segmented within each cluster using a cluster-specific superpixel density. This step leads to partial superpixel segmentations with different superpixel resolutions (blue, red, and green superpixel segmentations) that are combined and post-processed yielding the final superpixel segmentation. Note that the edge map is dilated for improved visibility. Input image taken from the SBD dataset [Gould et al., 2009].



(a) Boundary Recall **(b)** Undersegmentation Error **(c)** Overall Segmentation Qual.

Figure 6.3: Results of our edge-adaptive superpixel segmentation framework with SLIC [Achanta et al., 2012] using different edge detection methods: Sobel, Canny [Canny, 1986], gPb [Arbeláez et al., 2010], SE [Dollár and Zitnick, 2013, 2014], HED [Xie and Tu, 2015], and COB [Maninis et al., 2016, 2017]. For Sobel-based edge detection, we utilize the gradient magnitude based on both Sobel filters. The results are evaluated in terms of Boundary Recall (BR) (a), Undersegmentation Error (UE) (b), and the combined Overall Segmentation Quality (OSQ) (c) on the validation set of the BSD dataset [Martin et al., 2001].



Input image | Edge density | Clustering | Final superpixel segmentation

Figure 6.4: Intermediate results of our proposed edge-adaptive superpixel segmentation framework for three input images. The edge density (second column) is visualized as a map on top of the image and shows areas of high edge density, while low edge density regions are black. The third column presents the result of the k-means clustering based on the edge density and encodes the class membership of the pixels with different colors. Blue areas represent low edge density regions, red areas denote medium edge density regions, and green areas are high edge density regions. The final superpixel segmentation (fourth column) represents the combination of the superpixel segmentations with different superpixel resolutions according to the edge density. The color-coding of the superpixel segmentations is identical to the clustering. Input images taken from the SBD dataset [Gould et al., 2009].

the boundaries of the main objects while pruning uniform background areas (e.g., the sky) or highly textured areas (e.g., the water).

Based on edge density, we follow Gao et al. [2017] and apply k-means clustering to group pixels of similar edge density. Note that the areas may be disconnected, since strong edges may appear at several locations across the image (see third column in Fig. 6.4). The clusterings in Fig. 6.4 show that the cluster with the high edge density pixels (green cluster) mainly covers areas with numerous edges and fine details. Overall, each cluster represents pixels of similar edge density and will receive a unique superpixel resolution. This prevents a uniform distribution of superpixels across the image.

6.2.2 Adapting Superpixel Segmentations

Given the clustering, we estimate a superpixel resolution for each of the k clusters and segment each cluster individually using the estimated superpixel resolution. In general, high edge density should lead to a dense superpixel resolution and vice versa. To calculate the superpixel resolution per cluster, the user must provide the desired number of superpixels across the entire image n and the number of superpixels for the cluster with the lowest superpixel resolution n_1. Note that n_1 refers to the number of superpixels if the superpixel segmentation would cover the entire image. Since only limited image areas are segmented per cluster, the numbers of superpixels per cluster n_i with $i = 1, \ldots, k$ add up n based on the relative size of the cluster (a_i) in a weighted sum:

$$n = \sum_{i=1}^{k} a_i n_i. \tag{6.1}$$

The relative size of a cluster a_i is defined as the percentage of image pixels that are part of the cluster. Note that not fixing n_1 or the number of superpixels for the cluster with the highest superpixel resolution n_k leads to ambiguous results.

To calculate the number of superpixels for all clusters, we first transform Eq. 6.1 and replace n_i with an interpolation between n_1 and n_k:

$$n = \sum_{i=1}^{k} a_i \left(n_1 + (n_k - n_1) \frac{w(i)}{w(k)} \right). \tag{6.2}$$

The weight function $w(i)$ controls the interpolation and allows to exponentially increase the number of superpixels based on the average edge density per cluster e_i:

$$w(i) = \begin{cases} 1 - b^{\frac{e_i - e_1}{e_k - e_1}} & \text{if } b < 1, \\ b^{\frac{e_i - e_1}{e_k - e_1}} - 1 & \text{else.} \end{cases} \tag{6.3}$$

The exponential increase enables us to focus the generation of superpixels on the cluster with the highest edge density. The parameter b is optimized as described in Sec. 6.2.4.

Given Eq. 6.2, we can derive the number of superpixels for the cluster with the highest superpixel resolution n_k by simple transformations as

$$n_k = \frac{n_1 \sum_{i=1}^{k} a_i \left(1 - \frac{w(i)}{w(k)}\right) - n}{-\sum_{i=1}^{k} a_i \frac{w(i)}{w(k)}}. \tag{6.4}$$

(a) Joining **(b)** Step-wise overlay

Figure 6.5: Qualitative results of our edge-adaptive superpixel segmentation framework with SLIC [Achanta et al., 2012] using joining (a) and the step-wise overlay (b) to combine the superpixel segmentations from different clusters. Input images taken from the SBD dataset [Gould et al., 2009].

Since n_k in Eq. 6.2 is now fixed, we calculate the number of superpixels for the intermediate clusters as an interpolation between n_1 and n_k utilizing $w(i)$:

$$n_i = n_1 + (n_k - n_1)\frac{w(i)}{w(k)}. \tag{6.5}$$

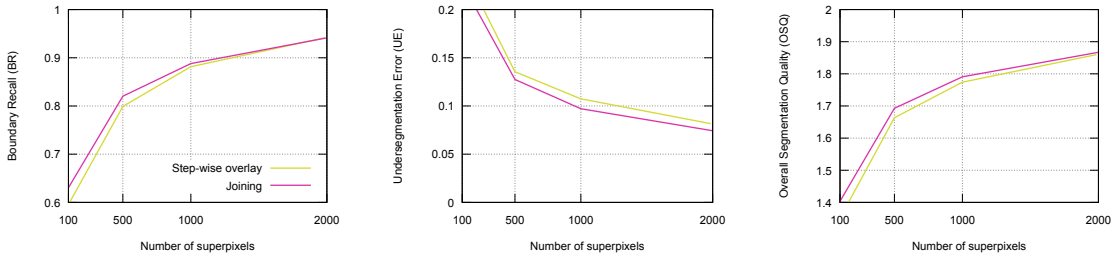
Based on the number of superpixels per cluster (n_i), we employ an arbitrary superpixel segmentation method per cluster using n_i. For simplicity, we apply the superpixel segmentation method to the entire image, while a parallel application to the clusters is possible for several methods like SLIC [Irving, 2016].

Overall, this stage leads to k superpixel segmentations with k different superpixel resolutions. This is highlighted in Fig. 6.2 (colored boxes) and the fourth column of Fig. 6.4 by the different colors of the partial superpixel segmentations. Green denotes the finest superpixel segmentation for the cluster with the highest superpixel resolution in both figures. In contrast, blue denotes the coarsest superpixel segmentation for the cluster with the lowest superpixel resolution. Note that Fig. 6.2 and Fig. 6.4 only show three clusters for visualization purposes. In general, we optimize k as described in Sec. 6.2.4.

6.2.3 Post-processing

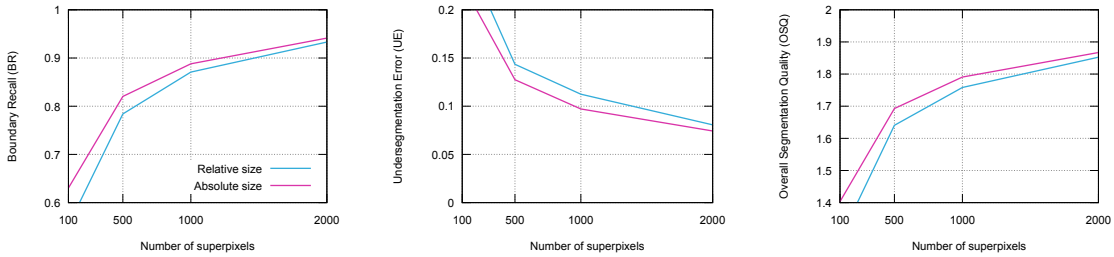
To generate a final superpixel segmentation, we join the superpixel segmentations of the different clusters. The joining assigns the superpixel label from the respective cluster to each pixel. As a result of the joining, artificial contours occur along the cluster boundaries in the final superpixel segmentation as visible in Fig. 6.5(a). However, this simple technique leads to better results than a step-wise overlay of superpixels as Fig. 6.6 indicates. The step-wise overlay starts with the coarsest superpixel segmentation and iteratively overlays the segmentation with complete superpixels from finer clusters. Hence, superpixels from finer clusters partially overlay coarser superpixels across the cluster boundaries. Figure 6.5 presents a qualitative comparison between both approaches. The artificial contours introduced by the joining are well visible, while the step-wise overlay leads to more oversegmentation in uniform areas, e.g., around the boat in the left scene.

As a result of the joining, several superpixels along the cluster boundaries are split into multiple parts. Moreover, most superpixel segmentation methods are unable to generate the desired number of superpixels precisely. To correct these artifacts, we first relabel all disconnected superpixel parts in the joined superpixel segmentation as independent superpixels. Subsequently, we apply a post-processing similar to SLIC and merge adjacent superpixels until



(a) Boundary Recall **(b)** Undersegmentation Error **(c)** Overall Segmentation Qual.

Figure 6.6: Quantitative results of our edge-adaptive superpixel segmentation framework with SLIC [Achanta et al., 2012] using joining and step-wise overlay to combine the superpixel segmentations from different clusters. The results are evaluated in terms of Boundary Recall (BR) (a), Undersegmentation Error (UE) (b), and the combined Overall Segmentation Quality (OSQ) (c) on the validation set of the BSD dataset [Martin et al., 2001].



(a) Boundary Recall **(b)** Undersegmentation Error **(c)** Overall Segmentation Qual.

Figure 6.7: Results of our edge-adaptive superpixel segmentation framework with SLIC [Achanta et al., 2012] using the absolute size and the cluster-specific relative size to select candidates for merging superpixels during post-processing. The results are evaluated in terms of Boundary Recall (BR) (a), Undersegmentation Error (UE) (b), and the combined Overall Segmentation Quality (OSQ) (c) on the validation set of the BSD dataset [Martin et al., 2001].

the desired number of superpixels n is reached. The merging process considers superpixels in order of increasing absolute size and fuses the superpixel with its most similar neighbor based on RGB color difference. This color-based merging differs from SLIC, where the neighbor is randomly selected. Using the absolute size to sort the superpixels will mainly merge superpixels from the cluster with the highest superpixel resolution. Therefore, we also consider merging based on relative size w.r.t. the average superpixel size of a cluster. However, as the results in Fig. 6.7 indicate, choosing the absolute size improves results.

The result of the post-processing is the final superpixel segmentation with n superpixels of various sizes (see bottom right of Fig. 6.2) that are non-uniformly distributed across the image. This leads to superpixel segmentations that adapt to the different levels of detail in the image and exhibit less oversegmentation. Note that we can apply our edge-adaptive superpixel segmentation framework to arbitrary superpixel segmentation methods that generate a desired number of superpixels.

6.2.4 Parameter Optimization

Our proposed edge-adaptive superpixel segmentation framework introduces six new parameters. We optimize them jointly with the parameters of the chosen superpixel segmentation method on the training set of a segmentation dataset in the framework of Stutz et al. [2018]. The

Table 6.1: Overview of the parameters in the proposed edge-adaptive superpixel segmentation framework with ranges of values for optimization.

Parameter	Range of values	Description
n	Given	Overall number of superpixels.
n_1	$\{\frac{1}{10}n, \frac{2}{10}n, \frac{3}{10}n, \frac{4}{10}n\}$	Number of superpixels for the cluster with the lowest superpixel resolution.
k	$\{2, 3, 4, 5, 6\}$	Number of clusters for k-means clustering.
b	$\{0.75, 2\}$	Base of the weighting function $w(i)$ in Eq. 6.3.
τ	$\{0.1, 0.2\}$	Threshold for binarizing the edge detection result.
σ	$\{2, 5, 10, 20\}$	Standard deviation for creating the edge density.

parameter optimization utilizes a grid search and assesses the segmentation quality based on Overall Segmentation Quality (OSQ) following Stutz et al. [2018]. The parameters to optimize based on a desired number of superpixels n are the number of superpixels for the cluster with the lowest superpixel resolution n_1, the number of clusters k, the threshold for binarizing the edge detection result τ, the standard deviation for creating the edge density σ, and the base b of the weighting function $w(i)$ in Eq. 6.3. Table 6.1 lists all parameters as well as the respective ranges of values. Note that we optimize all parameters for each combination of n, the superpixel segmentation method, and the dataset individually.

6.3 Superpixel Segmentation Results

First, we evaluate our edge-adaptive superpixel segmentation framework on the general superpixel segmentation task. This evaluation follows Stutz et al. [2018] utilizing the five challenging datasets BSD [Martin et al., 2001], SBD [Gould et al., 2009], Fash [Yamaguchi et al., 2012], NYU [Silberman et al., 2012], and SUN [Song et al., 2015] (see Sec. 2.1.2). We test our framework with the superpixel segmentation methods SLIC [Achanta et al., 2012] and ETPS [Yao et al., 2015]. Both methods lead to a large amount of oversegmentation and perform well in the benchmark of Stutz et al. [2018] as well as in SAM (see Sec. 5.4.3). We denote the edge-adaptive variations as *EA-SLIC* and *EA-ETPS*.

We compare EA-SLIC and EA-ETPS to original SLIC and ETPS, respectively. To show the upper limit of EA-SLIC and EA-ETPS, we replace the edge detection result with the ground truth edges and denote these variations as *GT-SLIC* and *GT-ETPS*. Additionally, we apply the saliency-based adaptation method of Gao et al. [2017] to images by replacing the edge maps in our framework with the saliency maps of Frintrop et al. [2015]. We coin these variations *Sal-SLIC* and *Sal-ETPS*. For the datasets NYU and SUN, which offer RGB-D information, we also compare EA-SLIC to the depth-based adaptation method *DASP* proposed by Weikersdorfer et al. [2013]. Since DASP uses a variation of SLIC, a comparison to EA-ETPS is not relevant. Note that DASP is the only method that utilizes depth information in this evaluation. A comparison to Kanezaki and Harada [2015] and Zhang et al. [2021] is impossible, since no code is publicly available. For a comparison of SLIC and ETPS with other superpixel segmentation methods, we refer to Stutz et al. [2018].

To assess the quality of the generated superpixel segmentations, we use Boundary Recall (BR), Undersegmentation Error (UE), and Oversegmentation Error (OE) (see Sec. 2.1.3) on the test sets of the five datasets. The parameter optimization for all variations of our edge-adaptive superpixel segmentation framework is conducted as described in Sec. 6.2.4. Similarly, we optimize the parameters of original SLIC and ETPS for every combination of the dataset and the number of superpixels individually.

In the following, we discuss the results of our edge-adaptive superpixel segmentation framework with SLIC (Sec. 6.3.1) and ETPS (Sec. 6.3.2) across all five datasets. Subsequently, we present the influence of the framework's parameters in Sec. 6.3.3. Note that we already presented ablation studies on design decisions in Sec. 6.2. For the results of the edge-adaptive superpixel segmentations with SAM, see Sec. 6.4.

6.3.1 Results using SLIC

Quantitative Results

Figure 6.8 presents the results of the SLIC-based methods and DASP in terms of BR, UE, and OE on the five datasets BSD, SBD, Fash, NYU, and SUN. Compared to SLIC and Sal-SLIC, our EA-SLIC outperforms both methods across all five datasets in terms of BR (left column in Fig. 6.8). The improvement of EA-SLIC over SLIC ranges from 5.1% on the BSD dataset to 9.1% on the SBD dataset. These improvements show that EA-SLIC captures more boundaries by focusing the generation of superpixels in relevant areas. The same effect is visible in the complex environments of the datasets NYU and SUN. If the detected edges in EA-SLIC perfectly represent the annotations (GT-SLIC), the improvements over SLIC are even up to 17.0% (SUN dataset). Comparing EA-SLIC to Sal-SLIC adapting Gao et al. [2017] reveals that EA-SLIC outperforms Sal-SLIC by 5.6% on average across the datasets in terms of BR. Note that Gao et al. [2017] originally developed their approach for point cloud data, not RGB images. On the datasets NYU and SUN, we can also compare EA-SLIC to the depth-adaptive method DASP. Similar to the findings in Stutz et al. [2018], DASP does not even reach the performance of SLIC in terms of BR, despite utilizing depth information.

Comparing the different methods in terms of UE across the five datasets (central column in Fig. 6.8) reveals mixed results. Except for the SBD dataset, SLIC exhibits a lower UE than EA-SLIC and Sal-SLIC. However, the absolute difference is low with a maximum difference of 0.013 for EA-SLIC and 0.018 for Sal-SLIC (both on the BSD dataset). Comparing SLIC to GT-SLIC utilizing ground truth edges shows a substantial improvement of 33.6% for GT-SLIC in terms of UE across the datasets. Hence, the performance of the edge-adaptive superpixel segmentations is strongly bound to the quality of the edge detection results. For instance, if the edge detection misses annotated edges, the area around the edge will be covered with large superpixels, leading to substantial undersegmentation errors. In contrast to the BR-based results, DASP produces competitive results in terms of UE.

Finally, the results in terms of OE across the five datasets (right column in Fig. 6.8) show the intended effect of reducing the oversegmentation in SLIC. Across all datasets, the OE improves by 4.5% on average from SLIC to EA-SLIC. Thus, the edge-adaptive superpixel segmentation framework decreases the effects of oversegmentation in uniform areas. Sal-SLIC leads to an improved OE as well (-1.5% on average). Providing ground truth edge detection results (GT-SLIC) strongly improves the OE (-12.5% on average) across the five datasets.

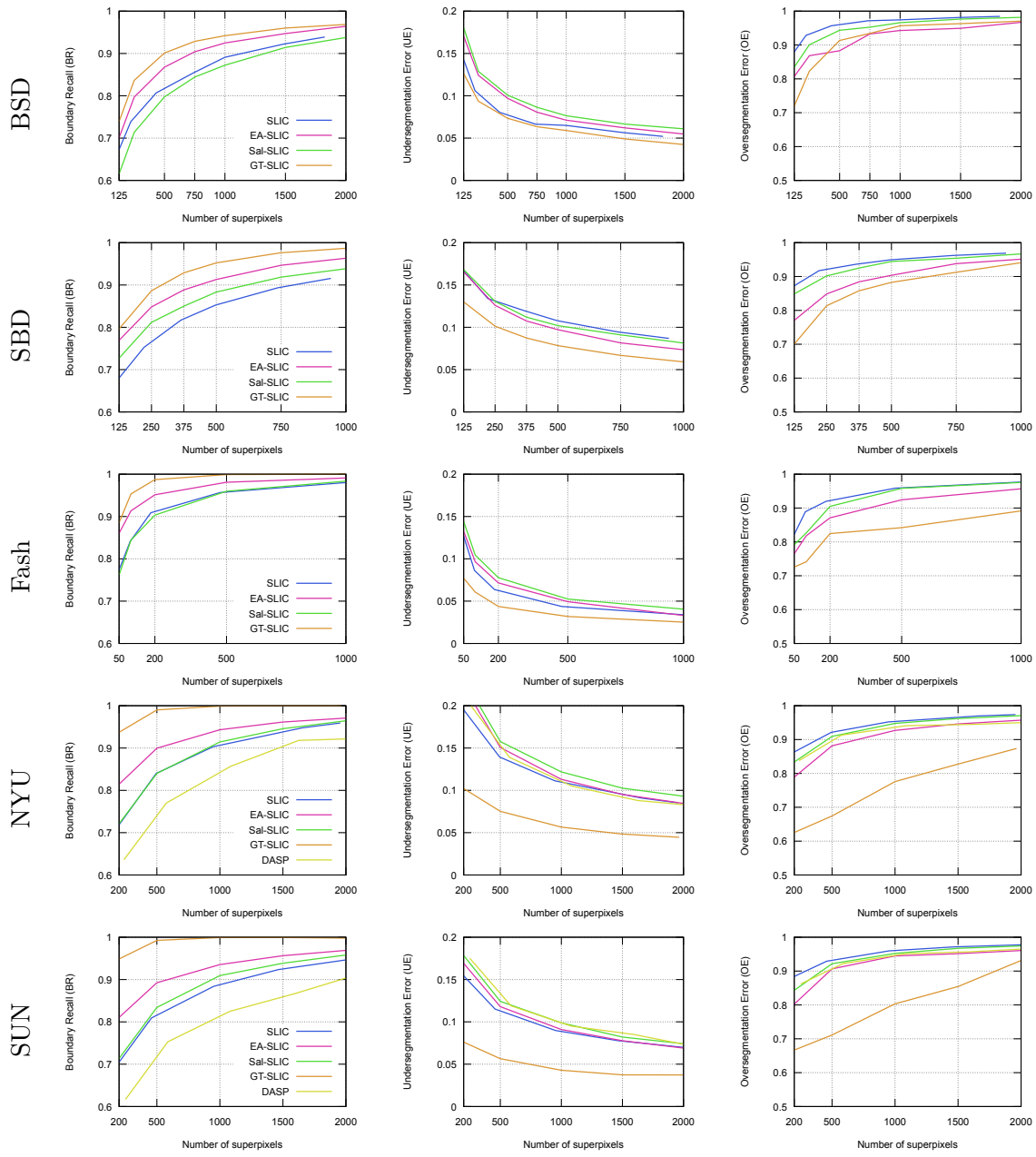


Figure 6.8: Quantitative superpixel segmentation results of our proposed EA-SLIC with SE edge detection results [Dollár and Zitnick, 2013, 2014], GT-SLIC utilizing ground truth edges, Sal-SLIC adapting Gao et al. [2017], and SLIC [Achanta et al., 2012]. The results in terms of Boundary Recall (BR, left), Undersegmentation Error (UE, center), and Oversegmentation Error (OE, right) are generated on the datasets BSD [Martin et al., 2001] (first row), SBD [Gould et al., 2009] (second row), Fash [Yamaguchi et al., 2012] (third row), NYU [Silberman et al., 2012] (fourth row), and SUN [Song et al., 2015] (fifth row).

Note that EA-SLIC outperforms GT-SLIC for 500 superpixels on the BSD dataset since GT-SLIC utilizes one of the multiple annotations per image (see Sec. 2.1.2) but still evaluates against all annotations of different granularity. DASP also reduces the OE compared to SLIC and leads to similar quantitative results as Sal-SLIC.

Overall, the quantitative results reveal an improved performance of EA-SLIC in terms of BR and a slight decrease in terms of UE compared to SLIC. Using the OSQ measure to combine BR and UE shows that EA-SLIC improves the segmentation quality of SLIC by 3.0%. Moreover, we showed the positive impact of our edge-adaptive framework in terms of OE without a loss in segmentation quality. Compared to the other adaptation methods (Sal-SLIC and DASP), EA-SLIC improves in almost all measures and datasets. Hence, our edge-adaptive superpixel segmentation framework is generally more suitable for adapting SLIC, although the results are strongly bound to the edge detection performance.

Qualitative Results

Figure 6.9 depicts qualitative results of SLIC, Sal-SLIC, and EA-SLIC on images from the datasets BSD, SBD, Fash, and NYU. These qualitative results allow a better understanding of the previously discussed quantitative results. In general, the results for EA-SLIC (third column) showcase the non-uniform distribution of superpixels across the image, which is different from SLIC (first column). For instance, in the first row, the sky is covered by three superpixels in the EA-SLIC result, while SLIC uses 27 superpixels. This effect is even visible for complex images with strongly textured areas like the grass in the third row.

Focusing on the segmentation quality, the results reveal improvements for EA-SLIC over SLIC and Sal-SLIC. In the example in the first row, the passengers are only segmented from the background using EA-SLIC. In contrast, SLIC uses many superpixels to oversegment sky, water, or mountain areas. The second row shows a typical problem of SLIC, where the arms and legs of the woman have a weak contrast w.r.t. the background, leading to missed boundaries. Since the SE edge detector captures those boundaries, EA-SLIC generates more superpixels here and segments the arms and legs. The final three examples show more textured or cluttered scenes revealing that EA-SLIC is robust to those effects as well. Hence, weak background edges or textured objects do not prevent EA-SLIC from capturing the relevant boundaries and outperforming SLIC. Examples are visible along the boundary of the tiger (third row) or the blanket (final row). Focusing on complex objects, the chairs in the fourth row are better captured by EA-SLIC as well due to the strong edge responses. In contrast, SLIC misses several of the chairs' boundaries. Thus, EA-SLIC has a strong positive effect on the segmentation quality as previously shown by the results in terms of BR.

Nevertheless, the quantitative results revealed a slight drop in terms of UE, which is also visible in some of the results in Fig. 6.9. Most prominently, the woman's belt in the example in the second row is not captured well by EA-SLIC. This results from missing edge responses on the upper boundary of the belt, leading to larger superpixels. Hence, undersegmentation errors are introduced that are not visible in SLIC. Another example is visible in the first row, where EA-SLIC does not properly capture the contour between the mountain parts due to missing edge responses. SLIC captures parts of the boundary reducing the undersegmentation errors. However, SLIC does not recognize the boundary explicitly but instead captures it by limiting the superpixel size.

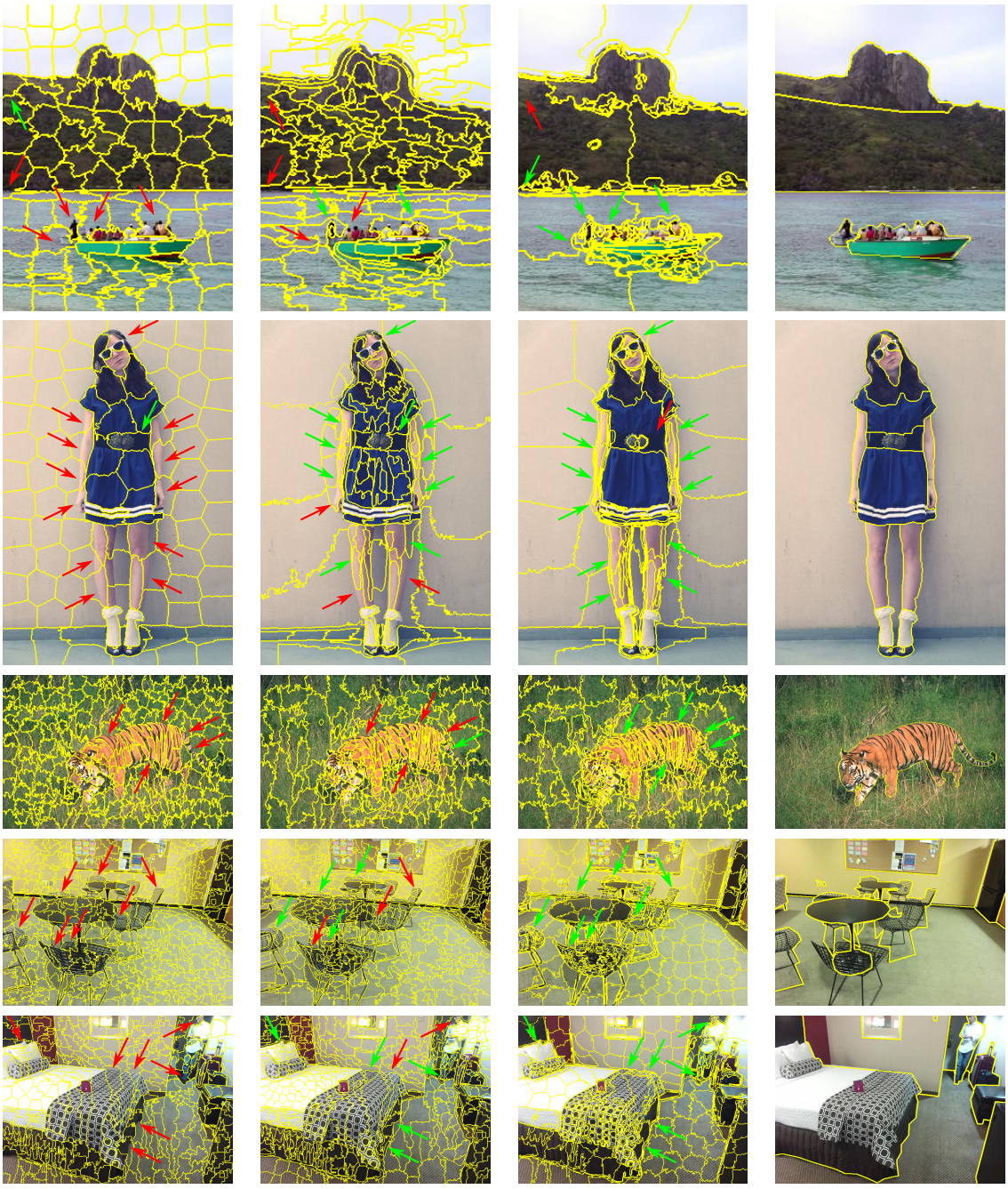


Figure 6.9: Qualitative superpixel segmentation results of SLIC [Achanta et al., 2012], Sal-SLIC adapting Gao et al. [2017], and the proposed EA-SLIC on images from the test sets of the datasets SBD [Gould et al., 2009] (first row), Fash [Yamaguchi et al., 2012] (second row), BSD [Martin et al., 2001] (third row), and SUN [Song et al., 2015] (fourth and fifth row). All results within a row contain a similar number of superpixels. The red arrows denote missed boundaries (undersegmentation errors), while the green arrows highlight the successful recognition of those boundaries. Input images and annotations taken from the datasets BSD [Martin et al., 2001], SBD [Gould et al., 2009], Fash [Yamaguchi et al., 2012], and SUN [Song et al., 2015].

Finally, we review the differences between EA-SLIC and Sal-SLIC. In general, Sal-SLIC shifts the focus of the superpixel creation toward salient regions. For instance, in the second row, EA-SLIC strongly focuses on the woman's outer boundaries and some boundaries of the garments. In contrast, Sal-SLIC focuses on the salient, dark blue dress that has a high contrast w.r.t. the background. Hence, the dress itself is segmented with many superpixels. As a result, the legs are covered with fewer superpixels, leading to missed boundaries. In general, Sal-SLIC does not necessarily focus on the relevant boundaries, leading to impaired results compared to EA-SLIC.

Overall, the qualitative results support and explain the quantitative results. Most prominently, the difference between SLIC and EA-SLIC in OE and BR is visible in the images. However, stronger undersegmentation errors induced by the focused processing and missed edges in EA-SLIC are also visible. Additionally, Fig. 6.9 reveals the different foci of Sal-SLIC and EA-SLIC.

6.3.2 Results using ETPS

Quantitative Results

Similar to the evaluation of EA-SLIC, Fig. 6.10 presents the results of ETPS, our EA-ETPS, Sal-ETPS, and GT-ETPS in terms of BR, UE, and OE on the five datasets BSD, SBD, Fash, NYU, and SUN. In terms of BR, EA-ETPS outperforms ETPS only on the most complex datasets NYU (+0.6%) and SUN (+1.7%). On the other datasets, the results are similar (Fash) or slightly worse (BSD and SBD). This behavior is different from EA-SLIC, where improvements in terms of BR occur across all datasets. However, all results in Fig. 6.10 are on a higher level compared to the SLIC-based results. Similar to SLIC-based results, GT-ETPS outperforms ETPS and EA-ETPS on three of the five datasets by up to 5.4%. Only on the BSD dataset with the multiple annotations and the simple Fash dataset no major improvement is visible for GT-ETPS compared to ETPS.

The results in terms of UE show that ETPS slightly outperforms EA-ETPS across most datasets (-0.004 on average), similar to the SLIC-based results. Overall, the segmentation quality in terms of OSQ slightly drops by 0.3% from ETPS to EA-ETPS across all datasets. Note that better edge detection results would improve the results of ETPS by up to 2.6% in terms of OSQ (GT-ETPS vs. ETPS across all datasets). Similar to the results of EA-SLIC, EA-ETPS improves the OE by 3.9% across all datasets compared to ETPS. For GT-ETPS, this improvement is even 8.6%. Hence, EA-ETPS and GT-ETPS successfully reduce the oversegmentation in ETPS. Moving from EA-ETPS and GT-ETPS to Sal-ETPS leads to degraded results across most datasets in terms of both BR and UE. Overall, the drop in OSQ compared to ETPS is 1.9% on average across all datasets. Similarly, for OE, the improvements (2.1% across all datasets) are below the level of EA-ETPS. Nevertheless, Sal-ETPS improves the OE compared to ETPS.

Overall, the results show that the segmentation quality is almost constant between ETPS and EA-ETPS, while Sal-ETPS reduces the segmentation quality. Given better edge detection results, even an improved segmentation quality is possible. Moreover, EA-ETPS successfully reduces the oversegmentation in ETPS. Hence, EA-ETPS is a good candidate for application in SAM due to the strong segmentation quality and the limited oversegmentation.

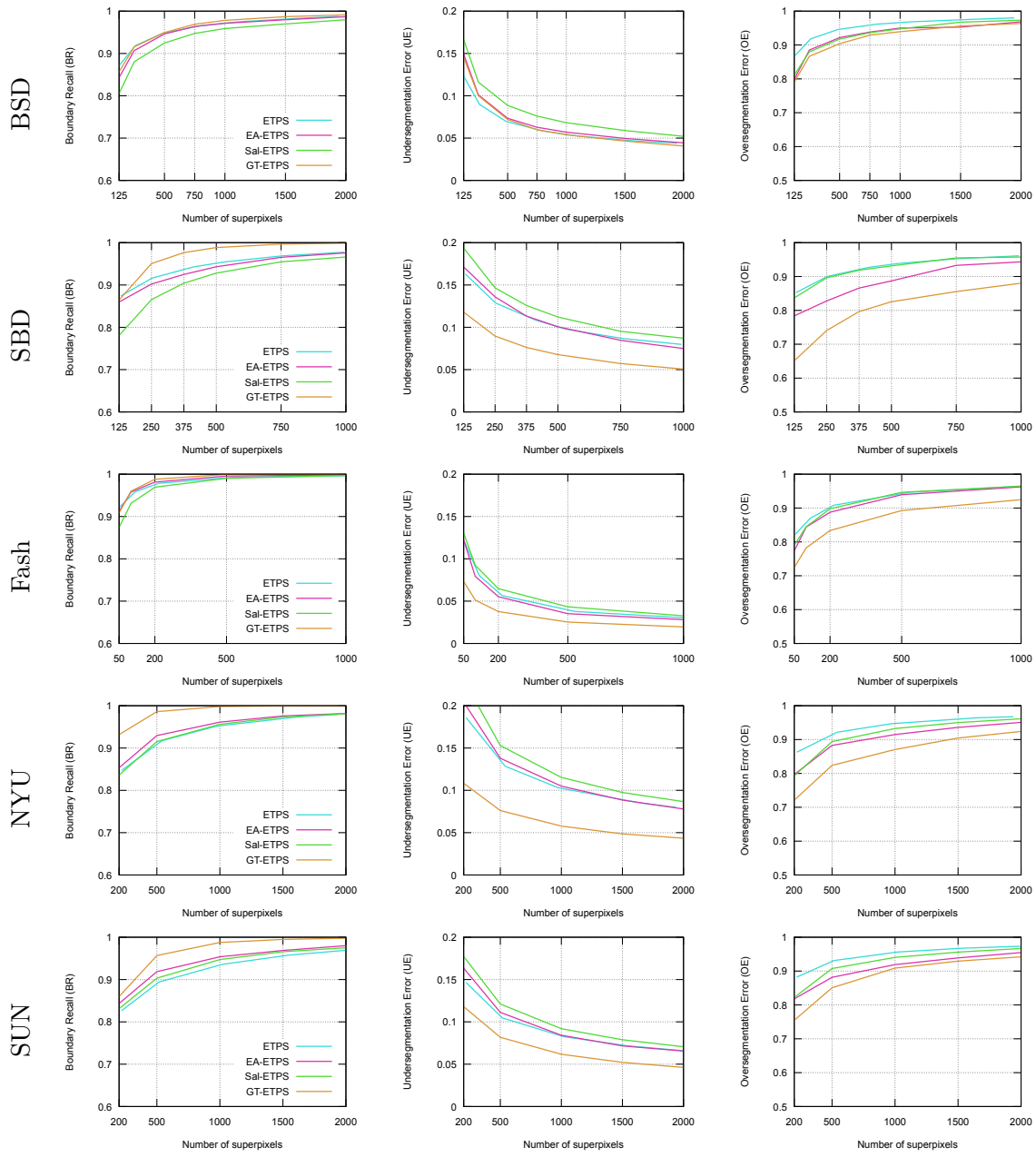


Figure 6.10: Quantitative superpixel segmentation results of our proposed EA-ETPS with SE edge detection results [Dollár and Zitnick, 2013, 2014], GT-ETPS utilizing ground truth edges, Sal-ETPS adapting Gao et al. [2017], and ETPS [Yao et al., 2015]. The results in terms of Boundary Recall (BR, left), Undersegmentation Error (UE, center), and Oversegmentation Error (OE, right) are generated on the datasets BSD [Martin et al., 2001] (first row), SBD [Gould et al., 2009] (second row), Fash [Yamaguchi et al., 2012] (third row), NYU [Silberman et al., 2012] (fourth row), and SUN [Song et al., 2015] (fifth row).

Qualitative Results

Figure 6.11 shows qualitative results of ETPS (first column), Sal-ETPS (second columns), and EA-ETPS (third column) on five images from the datasets BSD, Fash, NYU, and SUN. The general impression is similar compared to the SLIC-based results. EA-ETPS leads to a reduced oversegmentation error by non-uniformly distributing the superpixels across the image. However, the difference to ETPS is not as big as in the case of SLIC, since ETPS by itself leads to slightly less oversegmentation than SLIC. Still, the example in the first row highlights the strength of EA-ETPS to capture uniform areas with few superpixels by focusing the generation of superpixels on the image contours. Similar effects are visible in the other examples, e.g., on the street surface in the second row or the tabletops in the final row.

Improvements in terms of the segmentation quality are less evident as the quantitative results in terms of BR and UE implied. Additional boundaries are captured in low-contrast areas as visible in the first three examples. In the first image, the boundary of the woman's right arm is better captured by EA-ETPS since it generates more superpixels around the edges. The same holds true for the low-contrast boundaries in the examples in rows two and three (woman's left hand and sheep's horn). Moreover, small objects like the toothbrush or the tap in the fourth row are better captured by EA-ETPS since they are surrounded by a boundary, leading to an increased edge density. The final row shows an extreme case of low-contrast objects that only EA-ETPS captures. The chair legs are barely visible to the human eye. However, the edge detection results allow EA-ETPS to focus the generation of superpixels on these areas and capture the boundaries of the chair legs.

Besides these additionally captured boundaries, few typical undersegmentation errors are visible, similar to EA-SLIC. For instance, EA-ETPS misses parts of the upper boundary of the balustrade in the second row. Similarly, the right corner of the sink cabinet in row four is missed as well. These misses are similar to the behavior of EA-SLIC and arise from insufficient edge detection results.

Comparing the results of EA-ETPS and Sal-ETPS reveals several similarities. However, Sal-ETPS still misses details along the object boundaries by focusing on salient areas. For instance, Sal-ETPS focuses on the woman's hair in the first row that has a high contrast to the surrounding instead of focusing on the woman's right arm that ETPS misses. Similarly, the woman's skirt in the second row is salient, leading to a more detailed superpixel segmentation of this area. Hence, fewer superpixels cover details like the head or the low-contrast leg. Missing those low-contrast boundaries is similar to Sal-SLIC.

In general, the qualitative differences between ETPS and EA-ETPS are not as strong as in the case of SLIC and EA-SLIC. This is also supported by the quantitative results. Still, few typical improvements are visible in low-contrast areas or around small objects. Nevertheless, additional undersegmentation errors occur as well. EA-ETPS reduces the oversegmentation in ETPS as the non-uniform distribution of superpixels across the images demonstrates. This non-uniform distribution fits the quantitative results in terms of OE. Comparing the qualitative results of EA-ETPS and Sal-ETPS reveals the different focus of Sal-ETPS leads to missed low-contrast boundaries, similar to Sal-SLIC.

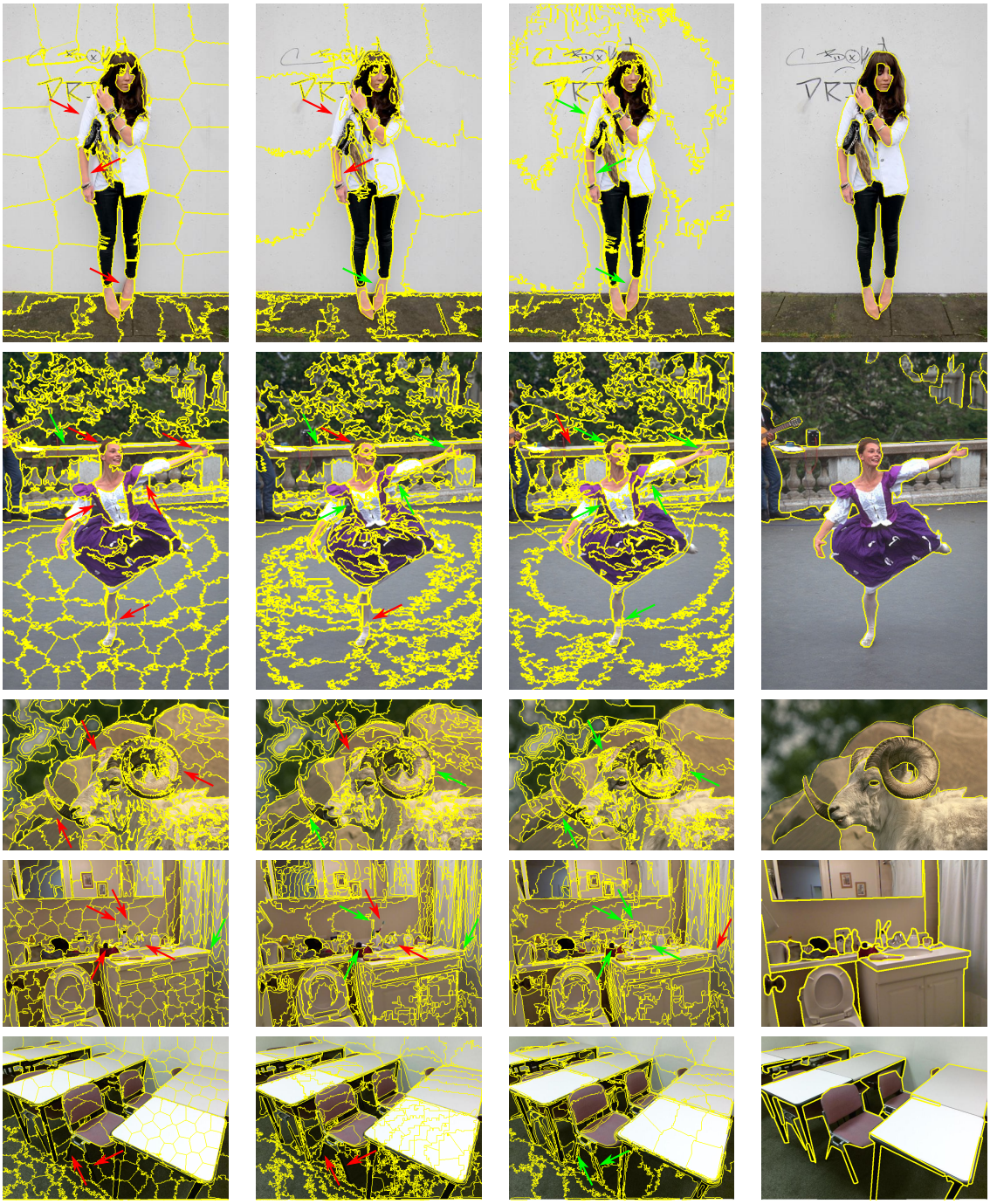


Figure 6.11: Qualitative superpixel segmentation results of ETPS [Yao et al., 2015], Sal-ETPS adapting Gao et al. [2017], and the proposed EA-ETPS on images from the test sets of the datasets Fash [Yamaguchi et al., 2012] (first row), BSD [Martin et al., 2001] (second and third row), NYU [Silberman et al., 2012] (fourth row), and SUN [Song et al., 2015] (fifth row). All results within a row contain a similar number of superpixels. The red arrows denote missed boundaries (undersegmentation errors), while the green arrows highlight the successful recognition of those boundaries. Input images and annotations taken from the datasets BSD [Martin et al., 2001], Fash [Yamaguchi et al., 2012], NYU [Silberman et al., 2012], and SUN [Song et al., 2015].

6.3.3 Influence of the Parameters

After presenting the results of EA-SLIC and EA-ETPS, we analyze the influence of the framework's parameters on the results. Our edge-adaptive superpixel segmentation framework has six parameters presented in Tab. 6.1. The parameters are the desired number of superpixels n, the number of superpixels for the cluster with the lowest superpixel resolution n_1, the number of clusters k, the base b of the weighting function in Eq. 6.3, the threshold τ for binarizing the edge detection results, and the standard deviation σ for creating the edge density. The subsequent analysis shows how sensitive the results are w.r.t. changes of the parameters. To assess the sensitivity to one parameter, we change this parameter in a pre-defined range and keep all other parameters constant. We set $n = 250$ in all experiments and apply EA-SLIC on the BSD validation dataset, similar to the ablation studies presented in Sec. 6.2. The results of EA-SLIC and the original SLIC in terms of BR, UE, and OSQ per parameter are visualized in Fig. 6.12.

First, we analyze the influence of n_1, the number of superpixels for the cluster with the lowest superpixel resolution. The parameter n_1 is essential in determining the number of superpixels in all clusters. We set n_1 to values between 10% and 40% of the overall number of superpixels (n). The results in the first row of Fig. 6.12 show that the BR is almost constant with varying n_1. However, the UE increases strongly for lower values of n_1. Since low values for n_1 lead to few superpixels in the cluster representing the background, missed boundaries lead to strong undersegmentation errors. Therefore, our edge-adaptive superpixel segmentation framework is sensitive to the value of n_1.

The second row in Fig. 6.12 shows the results for different numbers of clusters k. In terms of BR, many clusters (≥ 5) lead to impaired results. The reason for these results is the growing number of disconnected superpixels after joining the individual segmentations. As a result, the post-processing merges the disconnected superpixels and removes segmentation details, leading to undersegmentation errors. This drop also considerably impedes the OSQ for larger k's. Thus, the edge-adaptive superpixel segmentation framework is sensitive w.r.t. the choice of k.

Third, we analyze the influence of b, the base of the weighting function in Eq. 6.3. The weighting function controls the growth rate for the number of superpixels between the clusters. As the results in the third row of Fig. 6.12 imply, the differences in terms of BR, UE, and OSQ for values between 0.25 and 4 are minimal. Hence, the edge-adaptive superpixel segmentation framework is robust w.r.t. the choice of b.

To ignore weak edge detection results, we suppress results below a threshold τ. The fourth row in Fig. 6.12 shows the results for values between 0 and 0.4. A value of 0 implies that we use all edge detection results, while high values for τ lead to using high-confidence edge responses only. The results indicate that using all edge detection results and using only a few leads to subpar performances. In the case of $\tau = 0$, high edge densities exist across most of the image. In contrast, high values for τ lead to missing important contours in the image. This is visible from the decreasing BR and the increasing UE. Hence, the results drop below SLIC in terms of OSQ for values of τ above 0.2. Therefore, the edge-adaptive superpixel segmentation framework is sensitive w.r.t. the choice of τ.

Finally, we investigate the sensitivity of the edge-adaptive superpixel segmentation framework to σ. The parameter σ determines the size of the neighborhood for calculating the edge density. Hence, small values for σ lead to small neighborhoods and vice versa. As the results

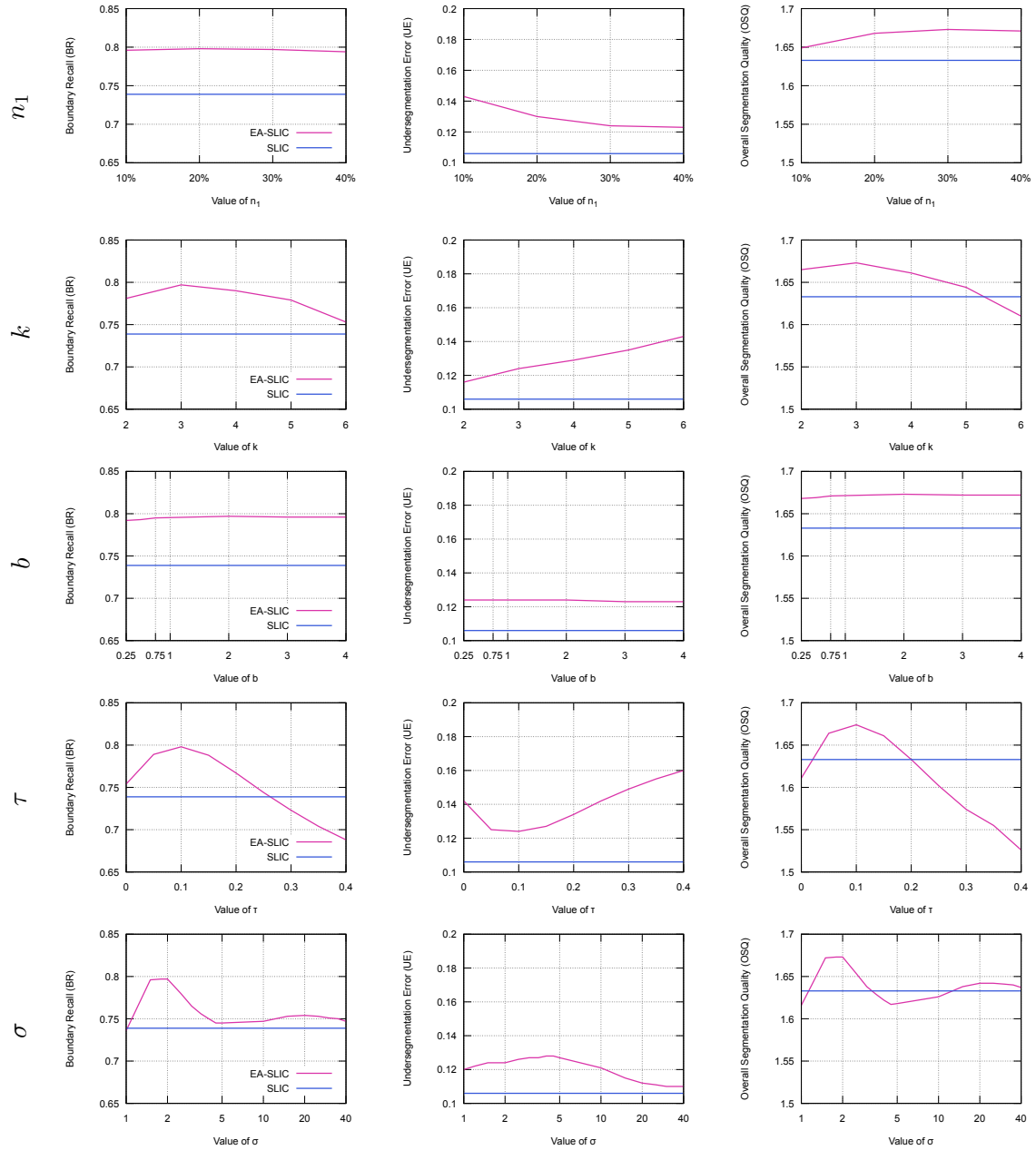


Figure 6.12: Quantitative superpixel segmentation results of SLIC and EA-SLIC with varying parameters on the BSD dataset with 250 superpixels. The different rows highlight the influence of EA-SLIC's parameters n_1 (first row), k (second row), b (third row), τ (fourth row), and σ (fifth row) on the results in terms of Boundary Recall (BR, left), Undersegmentation Error (UE, center), and Overall Segmentation Quality (OSQ, right).

in the final row of Fig. 6.12 show, our framework is sensitive to the choice of σ. The BR increases strongly for moderate values of σ, while the UE only drops at high levels of σ. Note that EA-SLIC converges to SLIC with large values for σ, since neighborhoods become similar across the image.

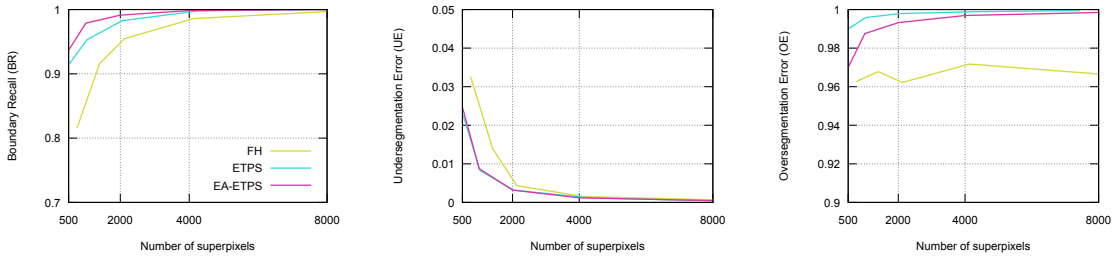
This analysis identified four important parameters for our edge-adaptive superpixel segmentation framework. The framework is sensitive to the choice of n_1, k, τ, and σ implying their great importance. A suboptimal choice of the four parameters can even lead to segmentation performances below the original SLIC baseline. Hence, for strong superpixel segmentation results, a thorough optimization of these parameters is necessary.

6.4 Object Proposal Generation Results

We now apply the edge-adaptive superpixel segmentations in our superpixel-based object proposal generation system SAM (see Ch. 5). As discussed in Sec. 5.4.3, SAM prefers superpixel segmentations with a low OE. The previous evaluation has shown that our edge-adaptive framework improves the OE for SLIC and ETPS while largely maintaining or improving the segmentation quality. Since ETPS leads to the second-best results when applied in SAM, we choose the EA-ETPS superpixel segmentation for application in SAM instead of FH. Applying the edge-adaptive superpixel segmentation framework to FH is impossible, since FH does not generate a desired number of superpixels. However, FH already limits the amount of oversegmentation by design.

The evaluation framework for the object proposal generation task stays unchanged compared to Sec. 5.4. Thus, we only evaluate pixel-precise proposals on the challenging LVIS test set with precise annotations, while training on the COCO training dataset for a fair comparison. To optimize EA-ETPS for the new data, we follow the same optimization as described in Sec. 6.2.4 on the LVIS validation set. We evaluate the overall object proposal generation results in terms of Average Recall (AR) across different numbers of proposals and object sizes. The AR assess how many objects are discovered and how well they are segmented (see Sec. 2.2.3). Moreover, we use the BR- and UE-based evaluation to assess the segmentation quality of the generated object proposals as introduced in Sec. 5.4. We compare the results of SAM utilizing EA-ETPS superpixel segmentations (*SAM+EA-ETPS*) to the original formulation of SAM with FH superpixels denoted as SAM+FH (see Ch. 5), AttentionMask (see Ch. 4), DeepMask [Pinheiro et al., 2015], SharpMask [Pinheiro et al., 2016], and FastMask [Hu et al., 2017a]. Besides those CNN-based systems, we also compare to MCG [Arbeláez et al., 2014; Pont-Tuset et al., 2017] and COB [Maninis et al., 2016, 2017], which do not utilize CNNs during proposal generation and therefore generate precise object proposals. Since we use the identical evaluation framework as in Sec. 5.4, the results of all methods except SAM+EA-ETPS stay unchanged.

We start our evaluation by assessing the superpixel segmentation results for FH, EA-ETPS, and ETPS on the LVIS validation dataset in Sec. 6.4.1. Moreover, Sec. 6.4.1 presents the results of SAM utilizing these superpixel segmentation on the LVIS validation dataset. Subsequently, in Sec. 6.4.2, we present the qualitative and quantitative results of SAM+EA-ETPS on the LVIS test dataset compared to other object proposal generation systems.



(a) Boundary Recall **(b)** Undersegmentation Error **(c)** Oversegmentation Error

Figure 6.13: Quantitative results of three superpixel segmentation methods on the LVIS validation set in terms of Boundary Recall (BR) (a), Undersegmentation Error (UE) (b), and Oversegmentation Error (OE) (c) across 500 to 8000 superpixels. The superpixel segmentation methods are FH [Felzenszwalb and Huttenlocher, 2004], ETPS [Yao et al., 2015], and our proposed EA-ETPS.

Table 6.2: Results of SAM (see Ch. 5) using three different superpixel segmentation methods to create superpixels. The superpixel segmentation methods are FH [Felzenszwalb and Huttenlocher, 2004], ETPS [Yao et al., 2015], and our proposed EA-ETPS. We use SAM with only five pyramid levels and generate the results on the LVIS validation set. AR denotes the Average Recall for the first 10, 100, or 1000 proposals. **Bold font** highlights the best results.

Superpixel segmentation	Number of superpixels	AR@10↑	AR@100↑	AR@1000↑
EA-ETPS	8000 – 500	**0.101**	**0.224**	**0.323**
FH (Ch. 5)	8000 – 500	0.100	0.221	0.320
ETPS	8000 – 500	0.097	0.214	0.311

6.4.1 Influence of the Superpixel Segmentations

Figure 6.13 presents the superpixel segmentation results of ETPS, EA-ETPS, and FH on the complex LVIS validation set in terms of BR, UE, and OE. The results show that the segmentation quality in terms of BR improves using EA-ETPS compared to ETPS. Hence, more object boundaries are captured while the UE stays constant. This is similar to the results of EA-ETPS and ETPS on the complex datasets NYU and SUN in Sec. 6.3.2. Moreover, as visible from the results in Fig. 6.13(c), the OE for EA-ETPS is reduced compared to ETPS. This is similar to the results in Sec. 6.3.2. Recall that this was the main goal of applying EA-ETPS in the context of SAM. However, the level of FH in terms of OE is not reached.

To assess the effect of the improved EA-ETPS superpixel segmentations, we utilize them in SAM. Table 6.2 present the results of SAM with EA-ETPS, ETPS, and FH superpixel segmentations on the LVIS validation set. The results show that EA-ETPS outperforms ETPS by 4.2% on average across the different AR values. Based on the findings from Ch. 5, we attribute this to the lower OE and higher BR results of the EA-ETPS superpixel segmentations. Comparing EA-ETPS and FH reveals that both are on a similar level with small improvements for EA-ETPS (+1.1% on average). Hence, despite the significant difference between EA-ETPS and FH in terms of BR, the FH superpixel segmentations are still competitive in this framework. This is mainly due to the still substantially lower OE. Overall, EA-ETPS improves the results of SAM on the LVIS validation set compared to the original ETPS.

Table 6.3: Results on the LVIS test dataset using pixel-precise segmentation mask proposals in terms of six Average Recall (AR) measures. AR^S, AR^M, and AR^L denote results on small, medium, and large objects. See Tab. 2.1 for details on the AR variations. The systems in the first part of the table do not use CNNs to generate proposals. The second part of the table contains systems based on a ResNet-50 backbone, while the systems in the third part utilize a ResNet-34 backbone. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Method	AR@10↑	AR@100↑	AR@1000↑	AR^S@100↑	AR^M@100↑	AR^L@100↑
MCG	0.048	0.131	0.237	0.031	0.204	0.462
COB	0.054	0.148	0.281	0.043	0.235	**0.477**
DeepMask	0.069	0.147	0.214	0.014	0.314	0.430
SharpMask	0.073	0.154	0.229	0.014	0.327	0.460
FastMask	0.069	0.161	0.256	0.055	0.296	0.386
AttentionMask (Ch. 4)	0.073	0.189	0.284	0.081	0.312	0.446
AttentionMask (Ch. 4)	0.076	0.185	0.271	0.083	0.305	0.423
SAM+FH (Ch. 5)	**0.092**	**0.206**	*0.290*	**0.094**	*0.335*	*0.471*
SAM+EA-ETPS	**0.092**	*0.204*	**0.293**	*0.093*	**0.337**	0.462

6.4.2 Results on the LVIS Dataset

Quantitative Results

The previous results indicated that EA-ETPS positively influences the segmentation quality in terms of BR, UE, and OE, which leads to improved results for SAM+EA-ETPS. Therefore, we present the results of SAM+EA-ETPS compared to other object proposal generation methods on the challenging LVIS test set in Tab. 6.3. The results show that SAM+EA-ETPS performs at a similar level compared to SAM+FH. Hence, SAM+EA-ETPS outperforms all CNN-based methods not using the superpixel-based refinement across all ARs. COB is the only system that outperforms SAM+EA-ETPS on large objects (+3.2%), similar to SAM+FH. Comparing SAM+EA-ETPS and SAM+FH further, reveals that the results are similar with a slight advantage for SAM+EA-ETPS in terms of AR@1000 and AR^M@100. However, the differences are not substantial and confirm the previous findings from Sec. 6.4.1 on the LVIS validation dataset.

Similar to the results in terms of AR, the results in Tab. 6.4 based on BR and UE show significant similarities between the segmentation quality of SAM+EA-ETPS and SAM+FH proposals. The average differences between the methods are 0.002 (BR) and 0.001 (UE). Similar to SAM+FH, SAM+EA-ETPS outperforms all CNN-based systems not utilizing our superpixel-based refinement. Compared to MCG and COB, which do not utilize CNNs to generate object proposals, SAM+EA-ETPS leads to mostly impaired results similar to SAM+FH. However, MCG and COB discover fewer objects leading to substantially lower results in terms of AR across all object sizes (see Tab. 6.3). Overall, the results in Tab. 6.3 and Tab. 6.4 show that our edge-adaptive superpixel segmentation framework allows SAM+EA-ETPS to perform on par with SAM+FH.

Qualitative Results

Following the quantitative results, we present qualitative results of AttentionMask, SAM+FH, and SAM+EA-ETPS on images of the LVIS test dataset displayed in Fig. 6.14. In general, the

Table 6.4: Detailed results on the LVIS test dataset using pixel-precise segmentation mask proposals in terms of Boundary Recall (BR) and Undersegmentation Error (UE). BR^S/UE^S, BR^M/UE^M, and BR^L/UE^L denote results on small, medium, and large objects. The systems in the first part of the table do not use CNNs to generate proposals. The second part of the table contains systems based on a ResNet-50 backbone, while the systems in the third part utilize a ResNet-34 backbone. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Method	BR↑	UE↓	BR^S↑	UE^S↓	BR^M↑	UE^M↓	BR^L↑	UE^L↓
MCG	*0.685*	0.073	0.833	**0.004**	0.709	0.023	*0.614*	*0.089*
COB	**0.734**	**0.059**	0.823	0.005	**0.738**	0.021	**0.686**	**0.068**
DeepMask	0.488	0.087	0.727	0.006	0.622	0.024	0.308	0.109
SharpMask	0.561	0.080	0.782	0.005	0.681	0.023	0.383	0.100
FastMask	0.510	0.084	0.794	0.006	0.622	0.023	0.318	0.107
AttentionMask (Ch. 4)	0.568	0.070	0.840	0.005	0.644	0.020	0.389	0.091
AttentionMask (Ch. 4)	0.547	0.075	0.832	0.005	0.637	0.020	0.356	0.099
SAM+FH (Ch. 5)	0.681	*0.068*	**0.864**	**0.004**	0.726	*0.019*	0.557	0.090
SAM+EA-ETPS	0.680	*0.068*	*0.862*	**0.004**	*0.727*	**0.018**	0.555	0.092

qualitative results are similar between SAM+EA-ETPS and SAM+FH. Hence, the SAM+EA-ETPS proposals adhere better to the object boundaries than the AttentionMask proposals as visible throughout all examples in Fig. 6.14. The similarities between the SAM+EA-ETPS and SAM+FH proposals are well visible from several examples in Fig. 6.14. In particular, the last four examples are either almost identical (airplane example in the fifth row) or vary only in minor details. For instance, both systems discover the tennis player in the seventh row. However, SAM+FH misses parts of the tennis player's boundary around the feet and the pants, while SAM+EA-ETPS misses parts of the right forearm. Similar examples exist along the boundaries of the clock in the sixth row and the zebra in the eighth row.

Apart from these similarities, a few examples show substantial differences between the SAM+EA-ETPS and SAM+FH proposals. For instance, SAM+FH does not properly capture the dog's head and tail in the first example. In the case of the head, this is due to an undersegmentation error as the detailed views of the proposal and the respective superpixel segmentation in Fig. 6.15 reveal. The SAM+EA-ETPS proposal adheres to the head's boundary since EA-ETPS produces many small superpixels along the boundary (see Fig. 6.15). A similar case is visible in the second row, where SAM+FH does not properly capture the bear's hind leg. Again, this is related to an undersegmentation error in the FH superpixel segmentation (see Fig. 6.15). EA-ETPS captures this low-contrast boundary as the detailed views in Fig. 6.15 show. Low-contrast boundaries are a general problem for FH as the example in the third row demonstrates. The low-contrast bear is only roughly discovered by the SAM+FH proposal similar to AttentionMask. In contrast, SAM+EA-ETPS successfully segments the snout and discriminates all three visible legs (see annotation in the final column). These improvements in low-contrast areas are also in line with the earlier findings in Sec. 6.3.2. Another typical problem in FH superpixel segmentations is the existence of antenna-like artifacts. Despite the post-processing in SAM, some artifacts remain as visible on the left-hand side of the fire hydrant in the fourth example. Such issues do not exist in EA-ETPS due to the compactness term in ETPS.

Overall, the qualitative results support the quantitative similarity between the proposals of SAM+EA-ETPS and SAM+FH. However, some improvements along low-contrast boundaries are visible

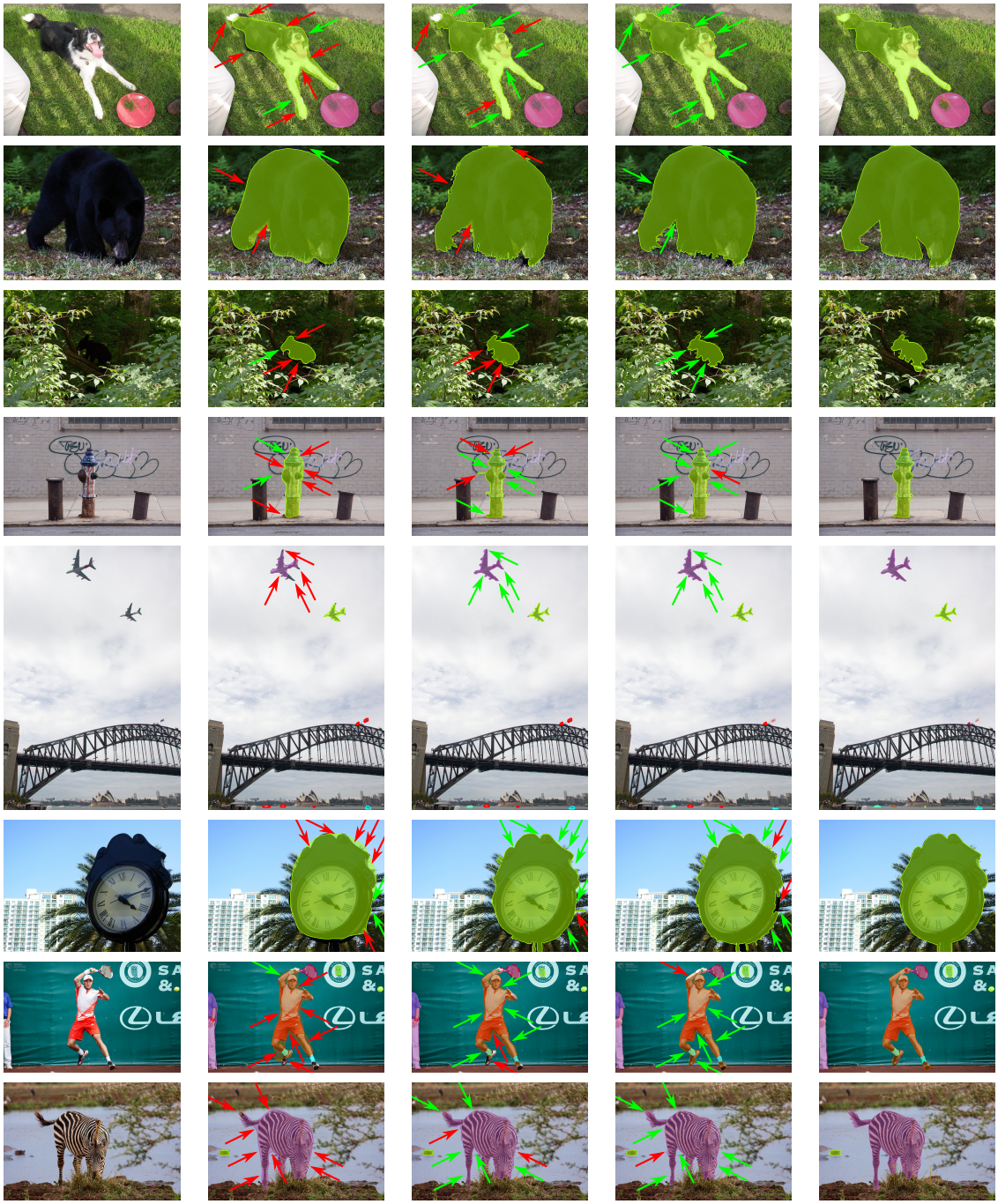
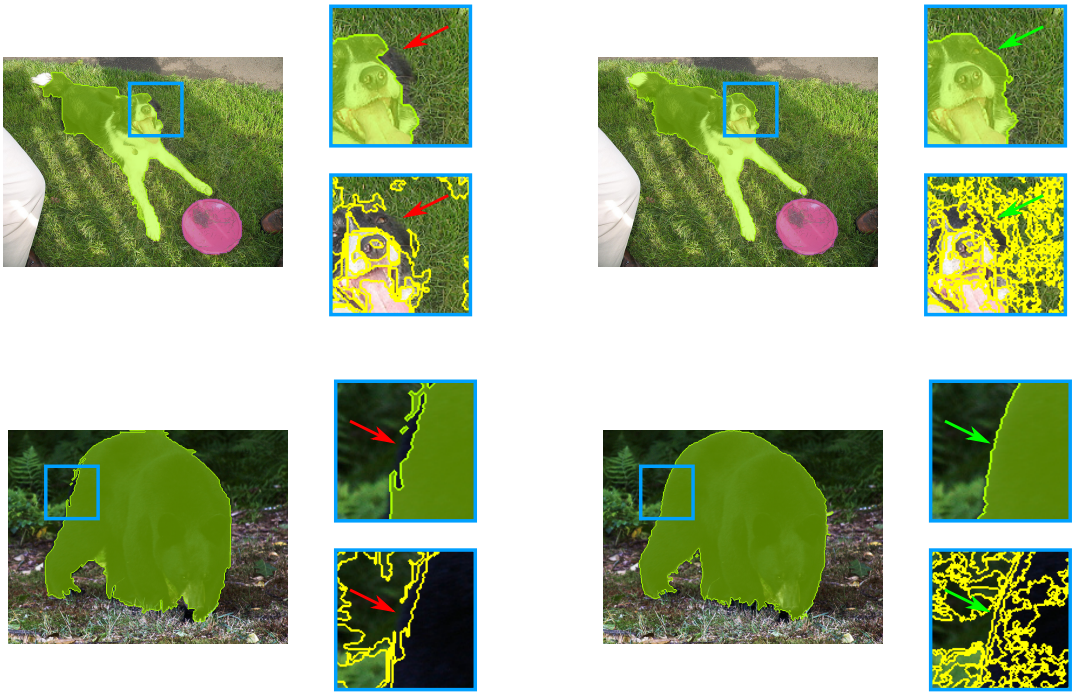


Figure 6.14: Qualitative results of AttentionMask based on a ResNet-50 (see Ch. 4), SAM+FH (see Ch. 5), and SAM+EA-ETPS on images of the LVIS test dataset. The arrows highlight prominent differences between the systems. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) is visualized per annotated object. Input images and annotations taken from the LVIS dataset [Gupta et al., 2019].



(a) SAM+FH **(b)** SAM+EA-ETPS

Figure 6.15: Detailed views of selected qualitative results of SAM+FH (a) and SAM+EA-ETPS (b). The blue boxes show upsampled crops of the proposal and the corresponding FH or EA-ETPS superpixel segmentation. Red arrows denote undersegmentation errors in the FH superpixel segmentations leading to imprecise proposals. Green arrows highlight the same areas in the EA-ETPS superpixel segmentations and the proposals. Input images and annotations taken from the LVIS dataset [Gupta et al., 2019]

6.5 Discussion

In this chapter, we introduced our novel edge-adaptive superpixel segmentation framework. Our framework adapts the uniform distribution of superpixels in superpixel segmentations to the different levels of detail in an image. As a result, the non-uniform distribution reduces the oversegmentation of the superpixel segmentations. To estimate the level of detail found in image areas, we calculate the edge density per pixel based on edge detection results. Subsequently, we define image regions with similar edge densities through clustering. By segmenting each cluster with an adapted superpixel resolution according to the edge density, we focus the generation of superpixels on image details. Hence, we reduce the oversegmentation of superpixel segmentation methods by adapting the distribution of superpixels.

Applying our edge-adaptive superpixel segmentation framework to SLIC and ETPS on five challenging datasets revealed that our framework reduces the oversegmentation (low OE). In contrast, the segmentation quality in terms of BR is either constant (ETPS) or even improves (SLIC). These results indicate the ability of our framework to capture relevant image details even in complex environments. Finally, the evaluation also showed that edge density is a better surrogate for the levels of detail in images than saliency or depth.

Utilizing the proposed EA-ETPS superpixel segmentations in our object proposal generation system SAM (see Ch. 5) leads to improvements over the results using ETPS on the complex LVIS dataset. This shows the importance of changing the segmentation style for improving

the results of SAM. Overall, the results of SAM+EA-ETPS are similar compared to the original SAM (SAM+FH) with improvements in low-contrast regions. However, EA-ETPS introduces five new parameters compared to the one parameter in FH. Hence, the optimization of EA-ETPS is more time-consuming compared to FH.

Overall, our novel edge-adaptive superpixel segmentation framework allows superpixel segmentation methods to reduce the oversegmentation while maintaining or improving the segmentation quality. The reduced oversegmentation improves the downstream object proposal generation with SAM compared to the original non-adaptive superpixel segmentations. Some remaining challenges of superpixel segmentations like the high number of parameters and the lack of incorporating high-level semantics will be addressed in the next chapter.

Chapter 7

DeepFH Superpixel Segmentation

Table of Contents

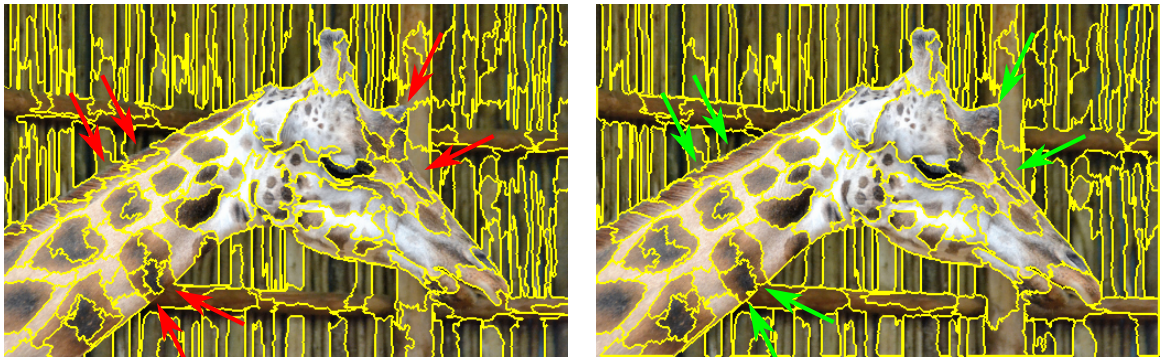
The edge-adaptive superpixel segmentation framework introduced in the previous chapter leads to improved superpixel segmentation results for SLIC [Achanta et al., 2012] and ETPS [Yao et al., 2015]. Moreover, the new EA-ETPS superpixel segmentations improve the object proposal generation results using SAM (see Ch. 5) compared to the original ETPS superpixel segmentations. However, SAM+EA-ETPS is unable to outperform the original SAM with FH superpixel segmentations consistently. Additionally, for optimizing the nine parameters[1] in EA-ETPS, a total of 8,640 combinations[2] were evaluated. Hence, the optimization of EA-ETPS for SAM is time-consuming. Different from superpixel segmentation methods like EA-ETPS or ETPS, FH [Felzenszwalb and Huttenlocher, 2004] (see Sec. 3.1.2) has only one parameter[3]. Hence, the optimization is less time-consuming. Additionally, FH superpixel segmentations are well-suited for application in SAM since they exhibit less oversegmentation than EA-ETPS and other superpixel segmentations.

However, due to the simplicity of FH, which merges pixels or groups of pixels based on the distance in RGB space, FH exhibit a lower segmentation quality compared to EA-ETPS or ETPS. This is also visible from the segmentation results on the LVIS dataset in Sec. 5.4.3. Additionally, since we aim to segment entire objects that may consist of different colors while sharing colors with adjacent background regions, utilizing only RGB information is suboptimal.

[1] Four parameters are inherited from ETPS.

[2] For computational reasons, we limit the number of values for γ_p and γ_b (see Eq. 3.1) as well as the iterations per hierarchy level to three.

[3] We use the implementation utilized in Stutz et al. [2018] that has a second parameter for the minimum size of each superpixel.



(a) Superpixel segmentation by FH **(b)** Superpixel segmentation by our DeepFH

Figure 7.1: Superpixel segmentation results of FH [Felzenszwalb and Huttenlocher, 2004] (a) and our proposed DeepFH superpixel segmentation method (b). The red arrows highlight areas where foreground and background have similar colors, leading to undersegmentation errors in the FH superpixel segmentation. In contrast, the proposed DeepFH superpixel segmentation method separates foreground and background in those areas (green arrows) based on learned features. Input image taken from the LVIS dataset [Gupta et al., 2019].

For instance, the marked areas in Fig. 7.1(a) highlight adjacent foreground and background pixels with similar color. FH fails to properly capture the giraffe's boundary in these areas using only RGB information, leading to undersegmentation errors. However, if the semantics of the pixels are known to FH, it would be able to capture the giraffe's boundary even in these areas. Therefore, FH needs semantically richer features to generate superpixel segmentations with better segmentation quality while maintaining a low oversegmentation. Such superpixel segmentations would subsequently also lead to improved object proposal generation results with SAM.

To enhance FH with semantically rich features, we propose the novel CNN-based *DeepFH* superpixel segmentation method. DeepFH augments the RGB features in FH with learned CNN-based per-pixel features representing high-level semantics. Subsequently, the original merging process of FH is applied to pixels or groups of pixels. We learn suitable features for superpixel segmentation using an encoder-decoder architecture that estimates pixel affinities as an auxiliary task similar to Tu et al. [2018]. However, in contrast to Tu et al. [2018], we do not use the affinities directly but utilize the latent features of the encoder-decoder. Since DeepFH employs the merging process of FH with improved features, it produces substantially less oversegmentation than EA-ETPS or existing CNN-based superpixel segmentation approaches. Hence, we generate superpixel segmentations based on high-level semantics that improve the segmentation quality compared to FH (see Fig. 7.1(b)) while limiting the amount of oversegmentation. Moreover, DeepFH introduces only one additional parameter for manual optimization compared to the original FH. Altogether, our CNN-based DeepFH superpixel segmentations will also boost the object proposal generation with SAM due to the favorable combination of improved segmentation quality and limited oversegmentation.

In this chapter, we introduce our novel DeepFH superpixel segmentation method based on our publication Wilms and Frintrop [2021]. First, in Sec 7.1, we describe the feature extraction network, the training setup, and the integration of the learned features into FH. Subsequently, we discuss the results of DeepFH on the superpixel segmentation task in Sec. 7.2. Section 7.3 presents the object proposal generation results of SAM with our new DeepFH superpixel segmentations. The chapter concludes with a discussion of the advantages and limitations of DeepFH in Sec. 7.4.

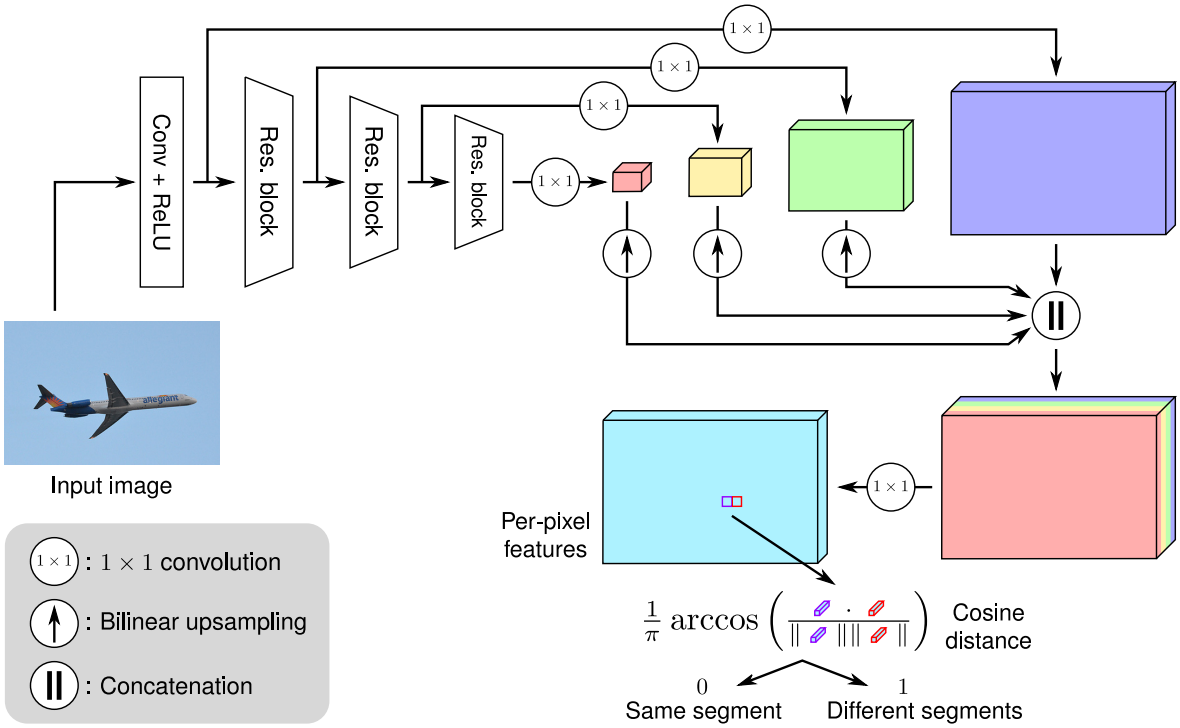


Figure 7.2: System figure of our proposed encoder-decoder-based feature extractor to generate per-pixel features in our DeepFH superpixel segmentation approach. First, the backbone network of the encoder generates features using residual blocks (res blocks). Subsequently, the different feature maps (different colors) are upsampled and concatenated. The final per-pixel features (turquoise box) are extracted from the concatenated feature map using a 1×1 convolution. During training, the cosine distance between each pair of adjacent pixels represents the pixel affinity that is used as an auxiliary output to train the network. Input image taken from the COCO dataset [Lin et al., 2014].

7.1 DeepFH Superpixels

This section presents our novel DeepFH superpixel segmentation approach utilizing learned CNN-based features for FH-style superpixel segmentations. DeepFH includes two new components compared to the original FH. First, a lightweight feature extractor utilizes features from different stages of an encoder-decoder network and generates a semantically rich feature map at input image resolution (see Fig. 7.2). The architecture of the feature extractor and the training strategy are described in Sec. 7.1.1. Second, in Sec. 7.1.2, we propose a new distance for assessing the pixel similarity in FH that combines RGB features and the new learned features. This allows a simple integration of the new features into FH. No further changes to FH are necessary.

7.1.1 Feature Extractor

The feature extractor is the core novelty of DeepFH since it generates semantically rich features. We utilize a simple encoder-decoder architecture to obtain such features and train the system in a pixel affinity framework.

Table 7.1: Architecture variations for the encoder (first part) and the decoder (second part) of our feature extractor. Ours denotes the proposed architecture for both parts. The evaluation is conducted w.r.t. the downstream task of object proposal generation using SAM. All results were generated on the LVIS validation set. AR denotes the Average Recall for the first 10, 100, or 1000 proposals. We use SAM with only five pyramid levels while the respective architecture is utilized for generating DeepFH superpixel segmentations. The chosen architecture is highlighted in **bold font**.

Architecture	AR@10↑	AR@100↑	AR@1000↑
Ours	**0.102**	**0.229**	**0.335**
ResNet-18	0.102	0.225	0.330
ResNet-34	0.102	0.226	0.332
Step-wise [Yang et al., 2020]	0.100	0.224	0.329

Architecture

The architecture of our feature extractor consists of a ResNet-based encoder with four stages and a simple concatenation-based decoder. This is a typical style of architecture for learning pixel affinity [Tu et al., 2018; Ahn and Kwak, 2018; Ahn et al., 2019]. As visible in Fig. 7.2, the encoder comprises the first four stages of a ResNet-18 (see Appendix A.2). However, only one residual block is utilized per stage, which leads to only nine trained layers in the encoder. Since we use a batch size of 1 during training (see below), we replace the batch normalization in the ResNet-18 with group normalization [Wu and He, 2018]. The encoder's lightweight design is more effective than the larger original ResNet-18 and ResNet-34 as the results in the first part of Tab. 7.1 show. These findings are in line with CNN-based superpixel segmentation systems that also use lightweight feature extractors with seven to ten layers [Tu et al., 2018; Jampani et al., 2018; Yang et al., 2020].

Based on the lightweight encoder, the decoder extracts features from each encoder stage by applying 1×1 convolutions. Thus, features from different semantic levels and spatial resolutions prevent heavily blurred feature maps. The extracted feature maps from the different stages are depicted as colored boxes in the top right of Fig. 7.2. After extraction, all feature maps are upsampled to input image resolution using bilinear interpolation. Subsequently, we concatenate the upsampled feature maps and generate the final 128D features per pixel (turquoise box in Fig. 7.2) by utilizing another 1×1 convolution. The concatenation is more favorable than a complex step-wise integration [Yang et al., 2020], as the results in the second part of Tab. 7.1 indicate. The final features are used in DeepFH to augment the simple RGB features.

Overall, our encoder-decoder-based feature extractor efficiently generates per-pixel features at input image resolution for generating high-quality superpixel segmentations.

Training

To train the proposed feature extractor, we utilize pixel affinity as an auxiliary task. Pixel affinity classifies adjacent pixels as part of the same segment or different segments, which is highly related to the superpixel segmentation task. We determine the pixel affinity results on

top of the extracted features (turquoise box in Fig. 7.2) by calculating the cosine distance $\delta_{\cos}$ between feature vectors $\mathbf{f}_i, \mathbf{f}_j \in \mathbb{R}^{128}$ of adjacent pixels $\mathbf{p}_i, \mathbf{p}_j \in \Omega$ as

$$\delta_{\cos}(\mathbf{f}_i, \mathbf{f}_j) = \frac{1}{\pi} \arccos\left(\frac{\mathbf{f}_i \cdot \mathbf{f}_j}{\|\mathbf{f}_i\|\|\mathbf{f}_j\|}\right). \tag{7.1}$$

The result of $\delta_{\cos}(\mathbf{f}_i, \mathbf{f}_j)$ is the result of the pixel affinity calculation. If adjacent pixels belong to the same segment, their distance should be 0, while adjacent pixels on different segments should have a distance of 1. We learn this task in a classification framework using binary cross entropy loss. Hence, while learning the affinity of adjacent pixels, the feature extractor learns useful features for generating superpixel segmentations. Note that backpropagation through the cosine distance is straightforward.

The ground truth for the pixel affinity calculation is generated based on the ground truth from segmentation datasets. In the case of object proposal generation datasets, we first overlay the different annotated objects to create an artificial ground truth segmentation. This segmentation contains all boundaries from all annotated objects. Given such a ground truth segmentation, the negative samples, i.e., pixels belonging to the same segment, will outnumber the positive samples, pixels belonging to different segments. To counter this imbalance, we use a negative sample mining strategy. Hence, for each positive sample, we randomly select up to three negative samples. Overall, given an input image, this process selects suitable pixels at the output level to train the feature extractor.

We train our feature extractor from scratch using the Adam optimizer [Kingma and Ba, 2015] with an initial learning rate of 0.01, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Training the network from scratch is similar to other CNN-based superpixel segmentation methods [Tu et al., 2018; Jampani et al., 2018; Yang et al., 2020].

7.1.2 Extension of FH

After training, the feature extractor generates semantically rich per-pixel features from unseen images. We use these semantically rich features, 128D vectors per pixel, to augment the feature distance computation in the original FH. In FH, the feature distance between adjacent pixels is defined as the Euclidean distance of the pixels' RGB values ($I_{\text{RGB}}(\mathbf{p}) : \Omega \to \mathbb{R}^3$). Hence, for pixels $\mathbf{p}_i, \mathbf{p}_j \in \Omega$ the original distance δ_{FH} is defined as

$$\delta_{\text{FH}}(\mathbf{p}_i, \mathbf{p}_j) = \|I_{\text{RGB}}(\mathbf{p}_i) - I_{\text{RGB}}(\mathbf{p}_j)\|_2. \tag{7.2}$$

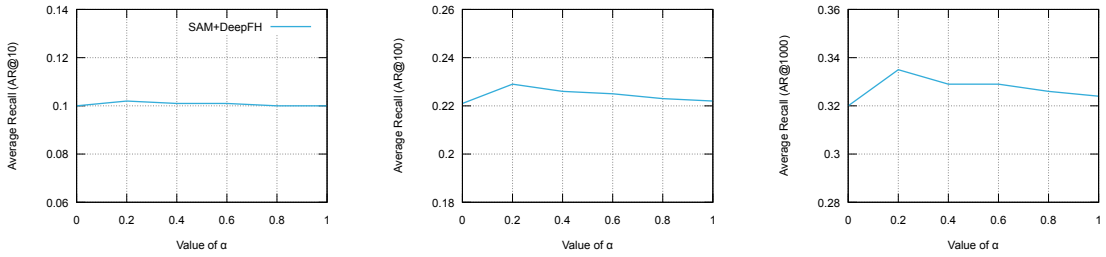
To augment this simple feature distance with the new learned features, we add the cosine distance between the extracted 128D feature vectors to δ_{FH}. Using the cosine distance instead of the Euclidean distance [Ahn and Kwak, 2018; Ahn et al., 2019] matches the training and fits the unnormalized high-dimensional feature vectors. This is further supported by the results in Tab. 7.2 that compares the use of Euclidean distance and cosine distance in DeepFH. Thus, our new combined feature distance δ_{DeepFH} is defined as

$$\delta_{\text{DeepFH}}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{f}_i, \mathbf{f}_j) = (1 - \alpha)\delta_{\text{FH}}(\mathbf{p}_i, \mathbf{p}_j) + \alpha\delta_{\cos}(\mathbf{f}_i, \mathbf{f}_j), \tag{7.3}$$

with the weight α to balance the influence of the two components. Note that we assume the individual RGB values in the interval of $[0, 1]$. This leads to two distances in the interval of $[0, 1]$, which simplifies the balancing. In our experiments, we set $\alpha = 0.2$ since it leads to

Table 7.2: Comparison of the cosine distance and the Euclidean distance for training the feature extractor and the calculation of δ_{DeepFH}. The evaluation is conducted w.r.t. the downstream task of object proposal generation with SAM. All results were generated on the LVIS validation set. AR denotes the Average Recall for the first 10, 100, or 1000 proposals. We use SAM with only five pyramid levels while the respective distance is utilized for generating DeepFH superpixel segmentations. The chosen distance is highlighted in **bold font**.

Distance	AR@10↑	AR@100↑	AR@1000↑
Cosine	**0.102**	**0.229**	**0.335**
Euclidean	0.101	0.223	0.327



(a) AR@10 **(b)** AR@100 **(c)** AR@1000

Figure 7.3: Results of SAM in terms of AR@10 (a), AR@100 (b), and AR@1000 (c) using DeepFH superpixel segmentations with varying α on the downstream object proposal generation task. All results were generated on the LVIS validation set. We use SAM with only five pyramid levels while the respective α value (x-axis) is utilized for generating DeepFH superpixel segmentations. AR denotes the Average Recall for the first 10, 100, or 1000 proposals.

optimized performance on the downstream object proposal generation task as the results in Fig. 7.3 indicate.

Based on the new feature distance δ_{DeepFH}, we adapt the initial pixel similarity in FH, while the remaining parts of FH stay unchanged. Thus, the merging of pixels in DeepFH is identical to the original FH (see Sec. 3.1.2). This preserves the segmentation style, leading to superpixel segmentations that exhibit limited oversegmentation while maintaining optimal results w.r.t. the feature distance [Felzenszwalb and Huttenlocher, 2004].

7.2 Superpixel Segmentation Results

To show the benefit of the learned features, we first present the results of DeepFH compared to the original FH on the superpixel segmentation task. Similar to Sec. 6.3 and Stutz et al. [2018], we evaluate on the five challenging datasets BSD [Martin et al., 2001], SBD [Gould et al., 2009], Fash [Yamaguchi et al., 2012], NYU [Silberman et al., 2012], and SUN [Song et al., 2015] (see Sec. 2.1.2). To assess the superpixel segmentations' quality, we use Boundary Recall (BR) and Undersegmentation Error (UE) for different numbers of superpixels as introduced in Sec. 2.1.3. Note that we do not compare DeepFH to other superpixel segmentation methods, since this chapter aims to improve FH to generate better object proposal generation results with SAM (see Sec. 7.3). A comparison of FH to other superpixel segmentation methods, which also allows an assessment of the DeepFH results w.r.t. those methods, is presented in Stutz et al. [2018].

The dataset-specific parameter optimization for FH follows Stutz et al. [2018]. However, we optimize the parameters for every number of superpixels individually. To choose the best set of parameters, we use the Overall Segmentation Quality (OSQ) (see Sec. 2.1.3) on the training data of the respective dataset. Since FH does not allow to set the desired number of superpixels explicitly, we vary the parameters to generate superpixel segmentations with various numbers of superpixels. Subsequently, we only consider results that deviate at most 10% from the desired number of superpixels similar to Sec. 5.3.1.

For DeepFH, the parameter optimization is conducted in two stages. First, we train the feature extractor on 70% of the respective datasets' training images while using the remaining training images for validation. Since only up to 238 images are available per dataset, we use the data augmentation strategy by Xie and Tu [2015] to increase the number of training samples by a factor of 96. The data augmentation strategy includes resizing, flipping, and rotating. Second, based on the trained feature extractor, we optimize the parameters of DeepFH using the same framework as for FH.

In the following sections, we first discuss the quantitative results of DeepFH and FH on the datasets BSD, SBD, Fash, NYU, and SUN in Sec. 7.2.1 before moving on to the qualitative results in Sec. 7.2.2.

7.2.1 Quantitative Results

Figure 7.4 presents the quantitative results for FH and our DeepFH on the superpixel segmentation task in terms of BR (left plots in Fig. 7.4) and UE (right plots in Fig. 7.4). On the datasets Fash (third row) and SBD (second row), DeepFH outperforms FH in terms of BR by 2.2% (Fash) and 2.0% (SBD) across the different numbers of superpixels. For the remaining datasets, the improvement is between 0.6% (BSD) and 1.2% (SUN). In terms of UE, the improvement is more significant with similar characteristics as for BR. For instance, DeepFH outperforms FH by 7.1% on the SBD dataset in terms of UE. Therefore, DeepFH generally captures more annotated boundaries than FH while reducing the leakage across the missed boundaries.

Despite improvements across all datasets, the improvements on the individual datasets vary as pointed out above. On the datasets Fash and SBD, DeepFH exhibits a stronger improvement compared to the other three datasets. These results are related to the image composition and the goal of the annotation process across the datasets. For instance, the images in the Fash dataset share a simple scene composition with a single person in front of varying backgrounds. The annotations are similar across the images since a limited set of garments and accessories are annotated. Given this limited variability in scene composition and annotations, DeepFH easily learns relevant features even from the few training images. For the SBD dataset, the scene and annotations vary more. However, objects like boats, cars, people, and cows are frequently reoccurring. This is different from the datasets NYU and SUN that feature a diverse set of objects and more complex scene compositions.

Overall, DeepFH leads to improved superpixel segmentation results compared to FH. However, the improvements vary between the datasets and are linked to the annotation style. Note that DeepFH only trained on 139 to 166 original training images per dataset. Therefore, better results are expected with more training data as in the case of the LVIS dataset (see Sec. 7.3.1).

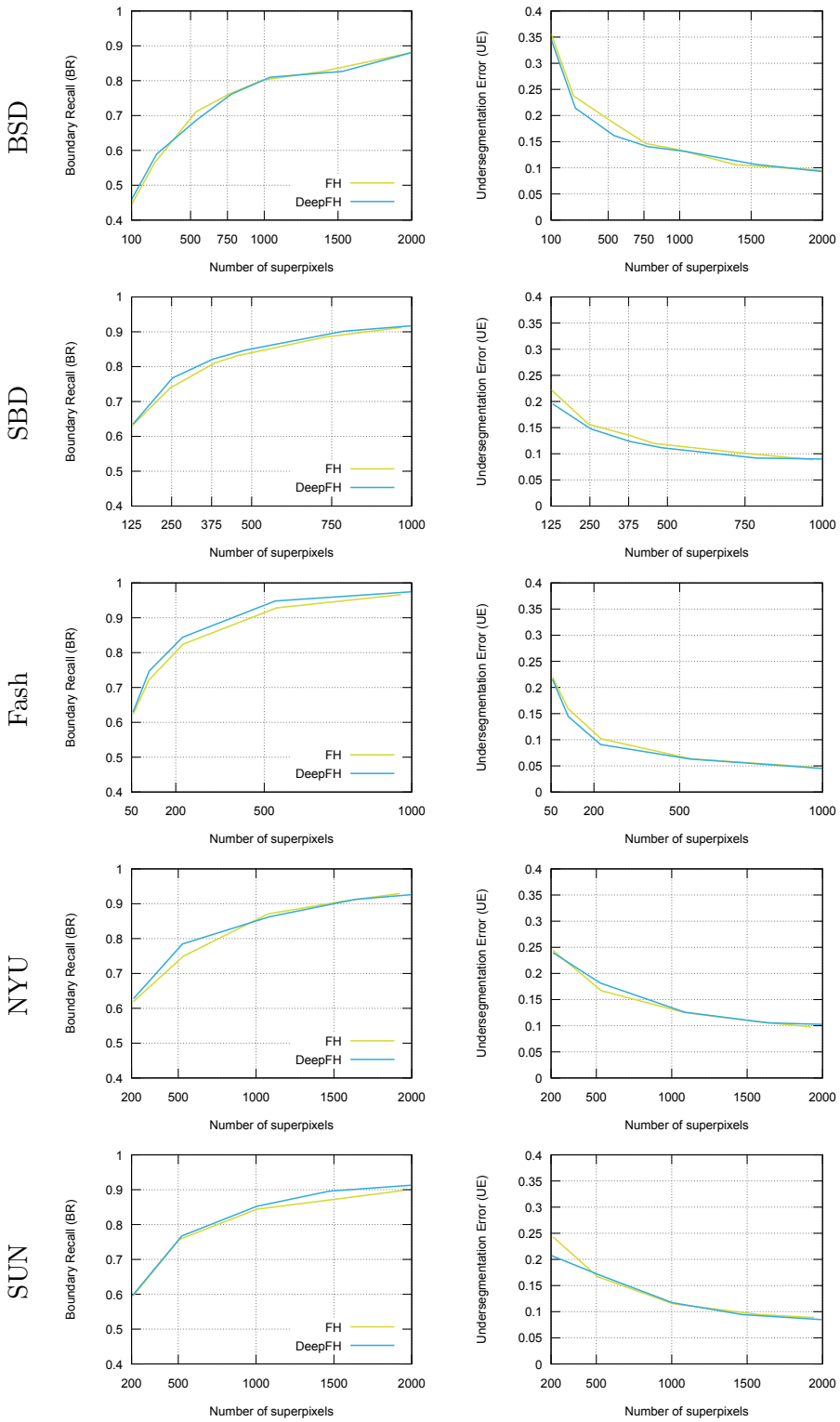


Figure 7.4: Quantitative superpixel segmentation results of our proposed DeepFH and FH [Felzenszwalb and Huttenlocher, 2004] on the datasets BSD [Martin et al., 2001] (first row), SBD [Gould et al., 2009] (second row), Fash [Yamaguchi et al., 2012] (third row), NYU [Silberman et al., 2012] (fourth row), and SUN [Song et al., 2015] (fifth row). The results are evaluated in terms of Boundary Recall (BR, left) and Undersegmentation Error (UE, right).

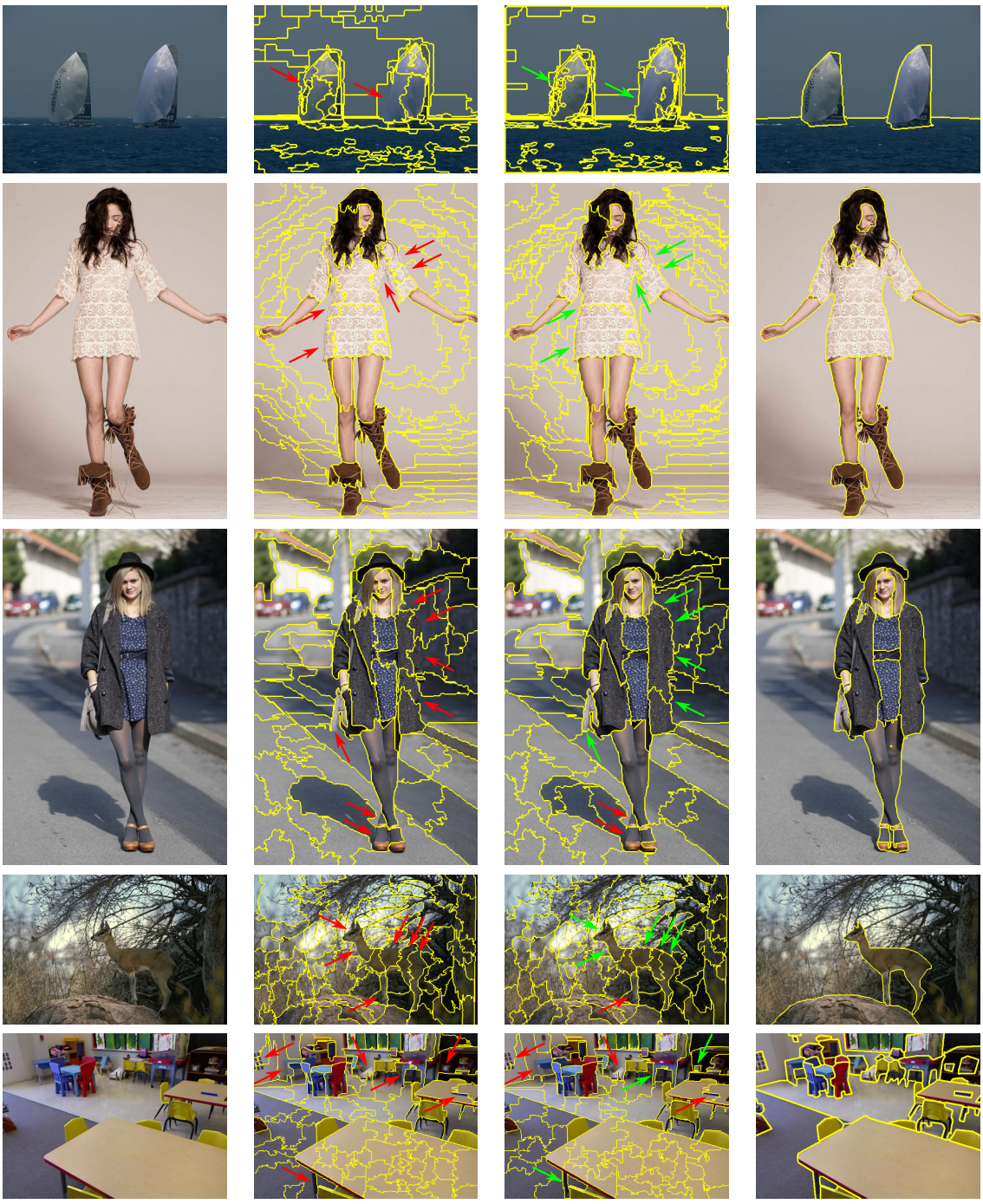


Figure 7.5: Qualitative superpixel segmentation results of FH [Felzenszwalb and Huttenlocher, 2004] and the proposed DeepFH on images from the test sets of the datasets SBD [Gould et al., 2009] (first row), Fash [Yamaguchi et al., 2012] (second and third row), BSD [Martin et al., 2001] (fourth row), and NYU [Silberman et al., 2012] (fifth row). All results within a row contain a similar number of superpixels. The red arrows denote missed boundaries (undersegmentation errors), while the green arrows highlight the successful recognition of those boundaries. Input images and annotations taken from the datasets SBD [Gould et al., 2009], Fash [Yamaguchi et al., 2012], BSD [Martin et al., 2001], and NYU [Silberman et al., 2012].

7.2.2 Qualitative Results

After discussing the quantitative results, we present qualitative results of DeepFH and FH for five images from the datasets BSD, SBD, Fash, and NYU in Fig. 7.5. The results show that the DeepFH and FH superpixel segmentations are similar and share the same segmentation style. This is important for the later application in SAM. The similarities between DeepFH and FH are not surprising, since DeepFH utilizes the general scheme of FH with semantically richer features.

Investigating the results in more detail reveals multiple differences between the DeepFH and the FH superpixel segmentations. For instance, DeepFH recognizes more boundaries in low-contrast areas. This behavior is visible from the results in the first row in Fig. 7.5, where FH misses a part of the sail in a low-contrast area (red arrows) that DeepFH captures. Similarly, in the second example, the woman's white dress exhibits a low contrast w.r.t. the background. Again, FH does not capture the boundary of the dress at several locations marked by the red arrows. DeepFH captures most of the boundary by utilizing CNN-based features that do not depend on color contrast. A similar setting is visible in the third row, where the boundary between the woman's jacket and the background is hard to see, even for humans. Still, DeepFH captures the entire boundary as highlighted by the green arrows, while FH misses most of it. An exception to this general improvement in low-contrast areas is the woman's right foot in the third row that neither system captures properly.

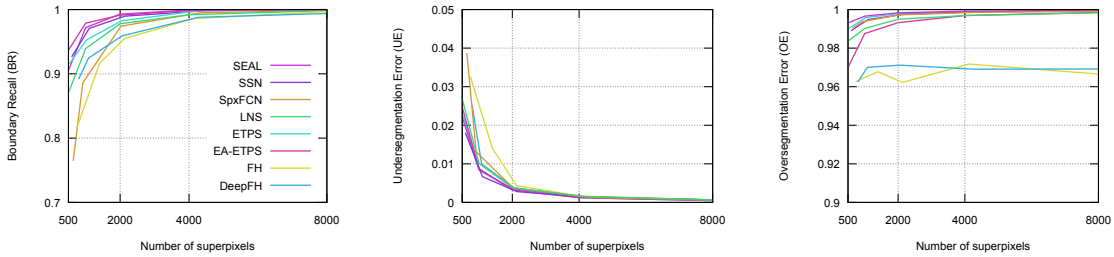
Besides low-contrast boundaries, highly textured areas pose another problem to the color-based FH. For instance, the antelope in the fourth row is not properly segmented from the highly textured background. DeepFH captures the boundary of the antelope except for one hoof since the learned features are not strongly distracted by the texture. However, more complex environments pose problems to both FH and DeepFH as the results in the final row highlight. Both systems miss several annotated boundaries in the cluttered indoor environment. Nevertheless, DeepFH captures more object boundaries than FH (see arrows).

Overall, the qualitative results showcase that the learned features in DeepFH improve the superpixel segmentations in environments with low contrast or strong texture. This confirms the advantage of the learned features compared to the simple RGB features used in FH. Moreover, the style of the FH segmentations is kept. However, complex environments present challenges to DeepFH as well.

7.3 Object Proposal Generation Results

The results on the superpixel segmentation task showed improvements of DeepFH in terms of BR and UE over FH while maintaining the segmentation style. To investigate if these improvements translate to better object proposals, we apply DeepFH in SAM.

For the evaluation of this combination on the object proposal generation task, we follow the evaluation protocol described in Sec. 5.4. Hence, we train all systems on the COCO training set and evaluate on the challenging LVIS test set featuring more precise annotations. To train the feature extractor in DeepFH, we use the LVIS training set, while optimizing the DeepFH parameters is conducted on the LVIS validation set. We use Average Recall (AR) in its different variations (see Sec. 2.2.3) to assess how many objects are discovered and how well they are segmented. A more detailed evaluation of the object proposals' segmentation quality is conducted using BR and UE as introduced in Sec. 5.4. Additionally, we



(a) Boundary Recall **(b)** Undersegmentation Error **(c)** Oversegmentation Error

Figure 7.6: Quantitative results of eight superpixel segmentation methods on the LVIS validation set in terms of Boundary Recall (BR) (a), Undersegmentation Error (UE) (b), and Oversegmentation Error (OE) (c) across 500 to 8000 superpixels. The superpixel segmentation methods are the CNN-based approaches SEAL [Tu et al., 2018], SSN [Jampani et al., 2018], SpxFCN [Yang et al., 2020], LNS [Zhu et al., 2021], and our proposed DeepFH as well as ETPS [Yao et al., 2015], EA-ETPS (see Ch. 6), and FH [Felzenszwalb and Huttenlocher, 2004], which do not utilize CNNs.

define a set of measures to assess the effectiveness of the superpixel utilization in SAM (see Sec. 7.3.3).

We compare SAM using DeepFH superpixel segmentations (*SAM+DeepFH*) with SAM+FH (see Ch. 5), SAM+EA-ETPS (see Ch. 6), AttentionMask (see Ch. 4), DeepMask [Pinheiro et al., 2015], SharpMask [Pinheiro et al., 2016], and FastMask [Hu et al., 2017a]. In addition to these CNN-based systems, we compare to MCG [Arbeláez et al., 2014; Pont-Tuset et al., 2017] and COB [Maninis et al., 2016, 2017], which do not utilize CNNs for object proposal generation, leading to precise object proposals. Since we use the identical evaluation framework as in Sec. 5.4 and Sec. 6.4, the results of all methods except SAM+DeepFH stay unchanged.

In the following, we first present the superpixel segmentation results of DeepFH, FH, and several other superpixel segmentation methods on the LVIS dataset in Sec. 7.3.1. Additionally, we discuss the effects of those superpixel segmentations on SAM. Section 7.3.2 presents the quantitative and qualitative results of SAM+DeepFH compared to other object proposal generation methods. Finally, in Sec. 7.3.3, we investigate the effective utilization of various superpixel segmentations in SAM.

7.3.1 Influence of the Superpixel Segmentations

First, we assess the superpixel segmentation quality of DeepFH and other methods on the challenging LVIS validation set in terms of BR, UE, and OE. Besides DeepFH, FH, EA-ETPS, and ETPS we also compare the four CNN-based superpixel segmentation methods SEAL [Tu et al., 2018], SSN [Jampani et al., 2018], SpxFCN [Yang et al., 2020], and LNS [Zhu et al., 2021] discussed in Sec. 3.1.3. From the results in terms of BR (see Fig. 7.6(a)) and UE (see Fig. 7.6(b)), it is visible that DeepFH outperforms FH in terms of segmentation quality. In terms of BR, the improvement is 2.2%, while the improvement in terms of UE is up to 17.6%. However, the other CNN-based superpixel segmentation methods as well as ETPS and EA-ETPS still mostly outperform DeepFH in BR and UE. In contrast, DeepFH and FH outperform all other methods in terms of OE (see Fig. 7.6(c)) by limiting the oversegmentation. Since the merging steps are identical in DeepFH and FH, the results in terms of OE are similar between both methods.

Table 7.3: Results of SAM (see Ch. 5) with eight different superpixel segmentation methods to create superpixels. The superpixel segmentation methods in the first part of the table utilize CNNs: SSN [Jampani et al., 2018], SEAL [Tu et al., 2018], SpxFCN [Yang et al., 2020], LNS [Zhu et al., 2021], and DeepFH. In contrast, the methods in the second part do not utilize CNNs: EA-ETPS (see Ch. 6), FH [Felzenszwalb and Huttenlocher, 2004], and ETPS [Yao et al., 2015]. We use SAM with only five pyramid levels and generate the results on the LVIS validation set. AR denotes the Average Recall for the first 10, 100, or 1000 proposals. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Superpixel segmentation	Number of superpixels	AR@10↑	AR@100↑	AR@1000↑
DeepFH	$8000 - 500$	**0.102**	**0.229**	**0.335**
SSN	$8000 - 500$	*0.101*	*0.224*	0.319
SEAL	$8000 - 500$	0.100	0.218	0.312
SpxFCN	$8000 - 500$	0.093	0.210	0.303
LNS	$8000 - 500$	0.095	0.210	0.296
EA-ETPS (Ch. 6)	$8000 - 500$	*0.101*	*0.224*	*0.323*
FH (Ch. 5)	$8000 - 500$	0.100	0.221	0.320
ETPS	$8000 - 500$	0.097	0.214	0.311

The combination of DeepFH's improved segmentation quality compared to FH and the similar results in terms of OE translate into strong object proposal generation results for SAM+DeepFH. As visible from Tab. 7.3, which shows the results of SAM with different superpixel segmentations on the LVIS validation dataset, DeepFH outperforms all other applied superpixel segmentations. Compared to the other CNN-based superpixel segmentations, the improvement is 6.8% on average. EA-ETPS and FH, which also exhibit a relatively low OE, are outperformed by 2.3% and 3.4%. These results confirm the findings from Sec. 5.4.3 that a low OE has a strong positive effect on the results of SAM and compensates for an impaired segmentation quality. Overall, DeepFH surpasses FH in terms of segmentation quality and outperforms all other tested superpixel segmentation methods when applied in SAM for object proposal generation.

7.3.2 Results on the LVIS Dataset

Quantitative Results

After showing the benefit of utilizing DeepFH in SAM, we compare the results of SAM+DeepFH to other object proposal generation systems on the complex LVIS test dataset. Table 7.4 shows the results in terms of AR that indicate a strong performance for SAM+DeepFH. In contrast to the previously introduced SAM+FH and SAM+EA-ETPS, SAM+DeepFH outperforms all other methods across all AR scores. This includes the highly precise COB proposals on large objects. Compared to AttentionMask utilizing a ResNet-50 backbone, which is the best overall system without superpixel-based refinement, the improvement is 12.7% for AR@100. Similar to the previous results on the LVIS validation set, SAM+DeepFH outperforms SAM+FH and SAM+EA-ETPS by 3.4% and 4.4% in terms of AR@100. Figure 7.7(a) presents a more detailed analysis of the AR-based results w.r.t. several numbers of proposals. Those results confirm the general improvements of SAM+DeepFH. Across all numbers of proposals,

Table 7.4: Results on the LVIS test dataset using pixel-precise segmentation mask proposals in terms of six Average Recall (AR) measures. AR^S, AR^M, and AR^L denote results on small, medium, and large objects. See Tab. 2.1 for details on the AR variations. The systems in the first part of the table do not use CNNs to generate proposals. The second part of the table contains systems based on a ResNet-50 backbone, while the systems in the third part utilize a ResNet-34 backbone. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Method	AR@10↑	AR@100↑	AR@1k↑	AR^S@100↑	AR^M@100↑	AR^L@100↑
MCG	0.048	0.131	0.237	0.031	0.204	0.462
COB	0.054	0.148	0.281	0.043	0.235	*0.477*
DeepMask	0.069	0.147	0.214	0.014	0.314	0.430
SharpMask	0.073	0.154	0.229	0.014	0.327	0.460
FastMask	0.069	0.161	0.256	0.055	0.296	0.386
AttentionMask (Ch. 4)	0.073	0.189	0.284	0.081	0.312	0.446
AttentionMask (Ch. 4)	0.076	0.185	0.271	0.083	0.305	0.423
SAM+FH (Ch. 5)	*0.092*	*0.206*	0.290	*0.094*	0.335	0.471
SAM+EA-ETPS (Ch. 6)	*0.092*	0.204	*0.293*	0.093	*0.337*	0.462
SAM+DeepFH	**0.094**	**0.213**	**0.304**	**0.098**	**0.349**	**0.480**

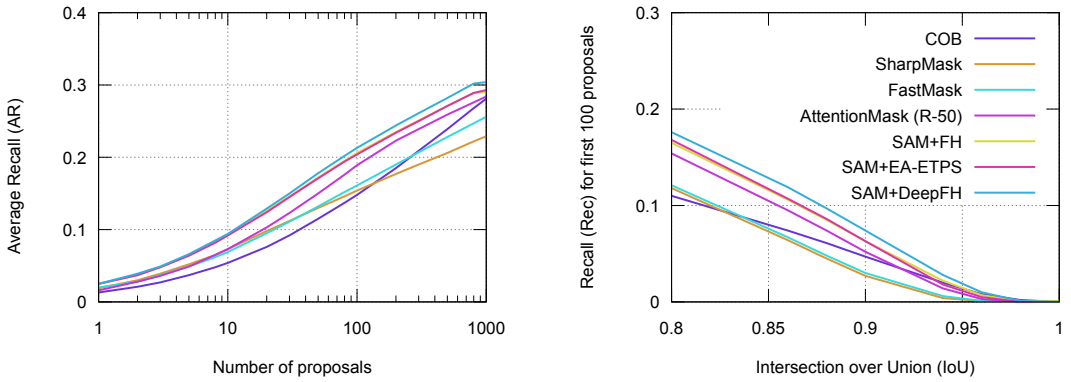
SAM+DeepFH outperforms all other methods in terms of AR by 3.3% (SAM+FH) to 54.0% (COB).

Table 7.5 presents the evaluation results in terms of BR and UE to analyze the boundary adherence of the SAM+DeepFH proposals. Apart from the COB proposals, SAM+DeepFH proposals adhere better to the annotated objects compared to all other methods. However, COB misses more objects than SAM+DeepFH (see Tab. 7.4). Compared to AttentionMask, the BR across all object sizes is improved by 23.2%, while the improvements w.r.t. SAM+FH and SAM+EA-ETPS are 2.8% and 2.9%. The same trend is visible for the results on the individual object sizes. This strong adherence to the object boundaries is also visible from the results in Fig. 7.7(b), which presents the Recall (Rec) for high IoU values. SAM+DeepFH constantly outperforms all other methods, including SAM+FH, SAM+EA-ETPS, and COB. The latter is notable since SAM+FH and SAM+EA-ETPS were unable to outperform COB on highly precise discoveries (IoU > 0.95).

Overall, the improved segmentation quality of DeepFH over FH and the improved OE compared to EA-ETPS translate into object proposal generation results for SAM+DeepFH that outperform all other systems.

Qualitative Results

To better understand the quantitative improvements of SAM+DeepFH, Fig. 7.8 depicts qualitative results of AttentionMask, SAM+FH, and SAM+DeepFH on images from the LVIS test set. Overall, the SAM+DeepFH proposals adhere better to the object boundaries than AttentionMask's proposals by utilizing the precise DeepFH superpixel segmentations. The improvements are visible along many object boundaries across all results in Fig. 7.8. For instance, the SAM+DeepFH proposals in the final two rows capture details like the smaller giraffe's ears and ossicones or the airplane's wingtips that are missed by the coarse AttentionMask proposals. In general, SAM+DeepFH shows improvements over AttentionMask similar to SAM+FH and SAM+EA-ETPS.



(a) **(b)**

Figure 7.7: Detailed quantitative results on the LVIS test set. (a): Average Recall (AR) across various numbers of object proposals for seven object proposal generation systems. (b): Recall (Rec) of seven object proposal generation systems for objects discovered with an IoU of at least 0.8. The results for AttentionMask are generated with a ResNet-50 backbone (R-50). Note that y-axes are truncated at 0.4 (a) and 0.3 (b) for improved visibility. The results for SAM+DeepFH and SAM+FH are almost at an identical level throughout the plots.

Table 7.5: Detailed results on the LVIS test dataset using pixel-precise segmentation mask proposals in terms of Boundary Recall (BR) and Undersegmentation Error (UE). BR^S/UE^S, BR^M/UE^M, and BR^L/UE^L denote results on small, medium, and large objects. The systems in the first part of the table do not use CNNs to generate proposals. The second part of the table contains systems based on a ResNet-50 backbone, while the systems in the third part utilize a ResNet-34 backbone. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Method	BR↑	UE↓	BR^S↑	UE^S↓	BR^M↑	UE^M↓	BR^L↑	UE^L↓
MCG	0.685	0.073	0.833	**0.004**	0.709	0.023	*0.614*	0.089
COB	**0.734**	**0.059**	0.823	0.005	**0.738**	0.021	**0.686**	**0.068**
DeepMask	0.488	0.087	0.727	0.006	0.622	0.024	0.308	0.109
SharpMask	0.561	0.080	0.782	0.005	0.681	0.023	0.383	0.100
FastMask	0.510	0.084	0.794	0.006	0.622	0.023	0.318	0.107
AttentionMask (Ch. 4)	0.568	0.070	0.840	0.005	0.644	0.020	0.389	0.091
AttentionMask (Ch. 4)	0.547	0.075	0.832	0.005	0.637	0.020	0.356	0.099
SAM+FH (Ch. 5)	0.681	0.068	*0.864*	**0.004**	0.726	0.019	0.557	0.090
SAM+EA-ETPS (Ch. 6)	0.680	0.068	0.862	**0.004**	0.727	**0.018**	0.555	0.092
SAM+DeepFH	*0.700*	*0.066*	**0.870**	**0.004**	*0.735*	**0.018**	0.590	*0.087*

Figure 7.8: Qualitative results of AttentionMask based on a ResNet-50 (see Ch. 4), SAM+FH (see Ch. 5), and SAM+DeepFH on images of the LVIS test dataset. The arrows highlight prominent differences between the systems. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) is visualized per annotated object. Input images and annotations taken from the LVIS dataset [Gupta et al., 2019].

Comparing the SAM+DeepFH proposals to the SAM+FH proposals, several differences in object boundary details are visible despite both systems' generally high boundary adherence. For instance, the dog's hind legs in the first row are only entirely captured by SAM+DeepFH. Typical areas of improvement for SAM+DeepFH compared to SAM+FH are low-contrast object boundaries. This is similar to the qualitative results on the superpixel segmentation task in Sec. 7.2.2. The second row presents a typical example where SAM+FH is unable to properly capture the woman's pants in the low-contrast environment. The low contrast leads to an undersegmentation error in the FH superpixel segmentation as the detailed view in Fig. 7.9 shows. Based on the learned features, DeepFH captures the pants' boundary (see Fig. 7.9), leading to a more precise object proposal. A similar case is visible in the third row. An undersegmentation error between the white pillow and the white blanket leads to an incomplete SAM+FH proposal for the blanket. DeepFH segments this area properly and generates an object proposal that captures the entire blanket. More examples of such behavior are visible along the front of the bed in the third row and the back of the bus in the fourth row.

(a) SAM+FH **(b)** SAM+DeepFH

Figure 7.9: Detailed views of selected qualitative results of SAM+FH (a) and SAM+DeepFH (b). The blue boxes show upsampled crops of the proposal and the corresponding FH or DeepFH superpixel segmentation. Red arrows denote undersegmentation errors in the FH superpixel segmentations leading to imprecise proposals. Green arrows highlight the same areas in the DeepFH superpixel segmentations and proposals. Input images and annotations taken from the LVIS dataset [Gupta et al., 2019].

Another type of undersegmentation error in FH leads to imprecise SAM+FH proposals in the fifth row of Fig. 7.8. As the detailed view in Fig. 7.9 shows, the FH superpixels do not capture the entire boundary along the right side of the smaller giraffe. This undersegmentation error in a highly textured area leads to the notch in the proposal. Similar effects are visible along the boundaries of both giraffes. As visible in Fig. 7.9, the learned features allow DeepFH to capture the complete boundary even in such highly textured environments, leading to the more precise SAM+DeepFH proposal.

Despite the generally high quality of the SAM+DeepFH proposals, few errors remain. For instance, individual superpixels are misclassified along the dog's stomach in the first row or the smaller giraffe's hind legs in the fifth row. These errors are due to the misclassification by the superpixel classifier and appear in SAM+FH as well (see Sec. 5.4.1). Hence, they are not related to the DeepFH superpixel segmentations.

In general, the qualitative results of SAM+DeepFH confirm the improved adherence to the object boundaries compared AttentionMask and SAM+FH. Moreover, the results show typical cases of improvement w.r.t. SAM+FH like low-contrast object boundaries or textured areas. These improvements highlight the benefit of the semantically rich learned features in DeepFH and are in line with the findings of the evaluation on the superpixel segmentation task in Sec. 7.2.2.

Table 7.6: Results of SAM (see Ch. 5) in terms of Average IoU (AVGIoU), Achievable IoU (AIoU), and Effectiveness (Eff) using eight superpixel segmentation methods to create superpixels. The superpixel segmentation methods in the first part of the table utilize CNNs: SSN [Jampani et al., 2018], SEAL [Tu et al., 2018], SpxFCN [Yang et al., 2020], LNS [Zhu et al., 2021], and DeepFH. In contrast, the methods in the second part do not utilize CNNs: EA-ETPS (see Ch. 6), FH [Felzenszwalb and Huttenlocher, 2004], and ETPS [Yao et al., 2015]. We use SAM with only five pyramid levels and generate the results on the LVIS validation set. **Bold font** highlights the best results, while *italic font* indicates the second-best results.

Superpixel segmentation	AVGIoU↑	AIoU↑	Eff↑
DeepFH	**0.455**	0.790	*57.6%*
SSN	0.443	*0.809*	54.7%
SEAL	0.438	0.777	56.3%
SpxFCN	0.442	0.781	56.6%
LNS	0.437	0.766	57.1%
EA-ETPS (Ch. 6)	*0.450*	**0.811**	55.5%
FH (Ch. 5)	0.441	0.766	**57.7%**
ETPS	0.424	0.799	53.0%

7.3.3 Superpixel Segmentation Effectiveness

After presenting the results of SAM+DeepFH, we evaluate the effective use of the superpixel segmentations in SAM+DeepFH and SAM in general. Specifically, we want to determine bottlenecks that limit the results of SAM. To this end, we introduce three new measures for evaluating object proposal generation results and superpixel segmentations. First, we propose *Average IoU* (AVGIoU) to evaluate object proposal generation results. AVGIoU is similar to ABO (see Sec. 2.2.3) and takes the IoU between each annotated object and the best fitting proposal. Subsequently, the resulting IoUs are averaged across all objects in a dataset. This assesses how well the annotated objects are segmented, similar to the evaluation in terms of BR and UE. Second, we introduce *Achievable IoU* (AIoU) to evaluate how well superpixel segmentations capture the boundaries of annotated objects. As discussed in Sec. 5.3.2, the superpixel segmentations do not capture all annotated objects' boundaries. Hence, the AIoU measures the IoU between the annotated object and the best combination of superpixels covering it. Similar to the AVGIoU, we calculate the AIoU per annotated object average across the dataset. This definition of AIoU allows us to subsequently estimate an upper limit for the AVGIoU given a perfect superpixel classifier in SAM. Finally, we combine both values as the ratio of AVGIoU and AIoU coined *Effectiveness* (Eff):

$$\text{Eff} = \frac{\text{AVGIoU}}{\text{AIoU}}. \tag{7.4}$$

Eff assesses how well SAM utilizes the superpixel segmentation capacity.

Table 7.6 shows the results of the evaluation of SAM utilizing various superpixel segmentations based on AVGIoU, AIoU, and Eff. The results in terms of AVGIoU roughly correlate with the AR-based results from Tab. 7.3 since AR and AVGIoU are closely related. Hence, DeepFH outperforms all other superpixel segmentations in terms of AVGIoU when applied in SAM. In terms of AIoU, EA-ETPS leads to the best results, followed by SSN. These results are in line with the results from Fig. 7.6, evaluating the superpixel segmentations based on BR

and UE. Therefore, EA-ETPS and SSN allow the best proposals given a perfect superpixel classifier in SAM. However, the current superpixel classifier does not utilize this potential. Therefore, FH and DeepFH exhibit the best Eff since they allow strong AVGIoU results given a mediocre AIoU. This highlights again that the superpixel classifier in SAM prefers superpixel segmentations with a low OE.

Another important finding from Tab. 7.6, which was not visible in previous results, is the generally low Eff across all superpixel segmentation methods. With an Eff between 53.0% and 57.7%, the high quality of the superpixel segmentations is not fully utilized by SAM. Hence, the superpixel segmentations allow better object proposals based on a more sophisticated superpixel classifier. Overall, according to the results in Tab. 7.6, the main remaining bottleneck in SAM is the quality of the superpixel classifier.

7.4 Discussion

This chapter presented DeepFH, our novel CNN-based extension of the traditional FH superpixel segmentation method. Integrating semantically rich CNN-based features allows DeepFH to overcome the limitations of FH, which is based on RGB features. Moreover, DeepFH produces superpixel segmentations with low oversegmentation like FH while adding only one manually tunable parameter. To learn CNN-based features for DeepFH, we proposed an encoder-decoder architecture in a pixel affinity framework. Subsequently, the latent features from the encoder-decoder are seamlessly integrated into FH by only adapting the initial feature distance computation. Hence, the merging process and the resulting properties of the superpixel segmentations are kept.

The results of DeepFH compared to FH on the superpixel segmentation task revealed improvements on most of the complex datasets in terms of BR and UE. Despite limited training data, DeepFH lifts the performance primarily on low-contrast contours, showcasing the advantage of semantically rich features. Still, the style of segmentations is similar to FH keeping the oversegmentation on a relatively low level. The relatively low oversegmentation distinguishes DeepFH from other superpixel segmentation methods like ETPS or EA-ETPS that produce stronger results in terms of BR and UE but lead to more oversegmentation.

Utilizing the new DeepFH superpixel segmentations in our object proposal generation system SAM (SAM+DeepFH) leads to the best result on the challenging LVIS dataset across all AR measures. Additionally, DeepFH outperforms all other superpixel segmentation methods tested with SAM. Compared to the original SAM (SAM+FH), improvements are mainly visible along low-contrast object boundaries, similar to the superpixel segmentation results. Finally, we introduced three new evaluation measures and showed that the main remaining bottleneck in SAM is the superpixel classifier. For FH and DeepFH, only 57.7% and 57.6% of the superpixel segmentation capacities are utilized by SAM.

Overall, our novel CNN-based DeepFH improves superpixel segmentation results compared to FH utilizing learned features. Additionally, DeepFH is the first superpixel segmentation method that combines learned CNN-based features with a low amount of oversegmentation. Despite not outperforming all existing superpixel segmentation methods in overall segmentation quality, the limited oversegmentation improves the results of subsequent applications. Hence, applying DeepFH in SAM leads to the best object proposal generation results among existing methods. Still, even better results are possible using a more sophisticated superpixel classifier.

Chapter 8

Extended Evaluation of Object Proposal Generation Systems

Table of Contents

The previous four chapters introduced our object proposal generation systems AttentionMask and SAM using different superpixel segmentations. We compared the results of our systems to other systems based on Average Recall (AR) as typically done in object proposal generation. Additionally, we calculated size-specific ARs for small, medium, and large annotated objects. The detailed evaluation w.r.t. object size led to the development of AttentionMask since prior systems exhibited subpar performances on small objects. Hence, detailed evaluations focusing on specific object properties are important to better understand challenges in current object proposal generation methods.

Similar to the size-specific evaluation, we investigate other object properties that impact the performance of object proposal generation systems in this chapter. This extends our previous evaluations and allows us to formulate challenges in object proposal generation that foster the development of new methods. The lack of such a detailed evaluation of challenging object properties beyond size is one of the limitations in object proposal generation discussed in the introduction. Closest to such an evaluation is the work of Hosang et al. [2015], who evaluate the robustness of pre-CNN object proposal generation methods under image transformations. However, the results do not identify challenging object properties to guide object proposal generation research.

To identify such challenging settings, we define six new properties for annotated objects that strongly influence object proposal generation results. First, we measure the color contrast of the object w.r.t. its surrounding and the textureness of an object to capture the

visual appearance. Additionally, we characterize the shape of the annotated objects using compactness, convexity, and eccentricity. Finally, we investigate the influence of the object classes on the results. Based on these properties, we generate new property-specific ARs similar to the existing size-specific ARs and evaluate modern object proposal generation systems in more detail.

Overall, we extensively investigate the results of nine object proposal generation systems on the challenging COCO and LVIS datasets w.r.t. six object properties. In the following, we first introduce the six object properties and the general setup of the evaluation in Sec. 8.1. In Sec. 8.2, we present the results of the nine systems w.r.t. each object property and highlight links between different properties. To conclude this chapter, we summarize and discuss the main findings in Sec. 8.3, stating four challenges in current object proposal generation.

8.1 Experiment Setup

This section presents the setup of our extended evaluation. First, we define the object properties used to generate property-specific results in Sec. 8.1.1. Subsequently, Sec. 8.1.2 introduces the dataset selection and the evaluated object proposal generation methods. Finally, in Sec. 8.1.3, we present the evaluation measures to assess the performance of the object proposal generation methods w.r.t. to the previously defined object properties. Moreover, we discuss how to estimate the dependency between different object properties.

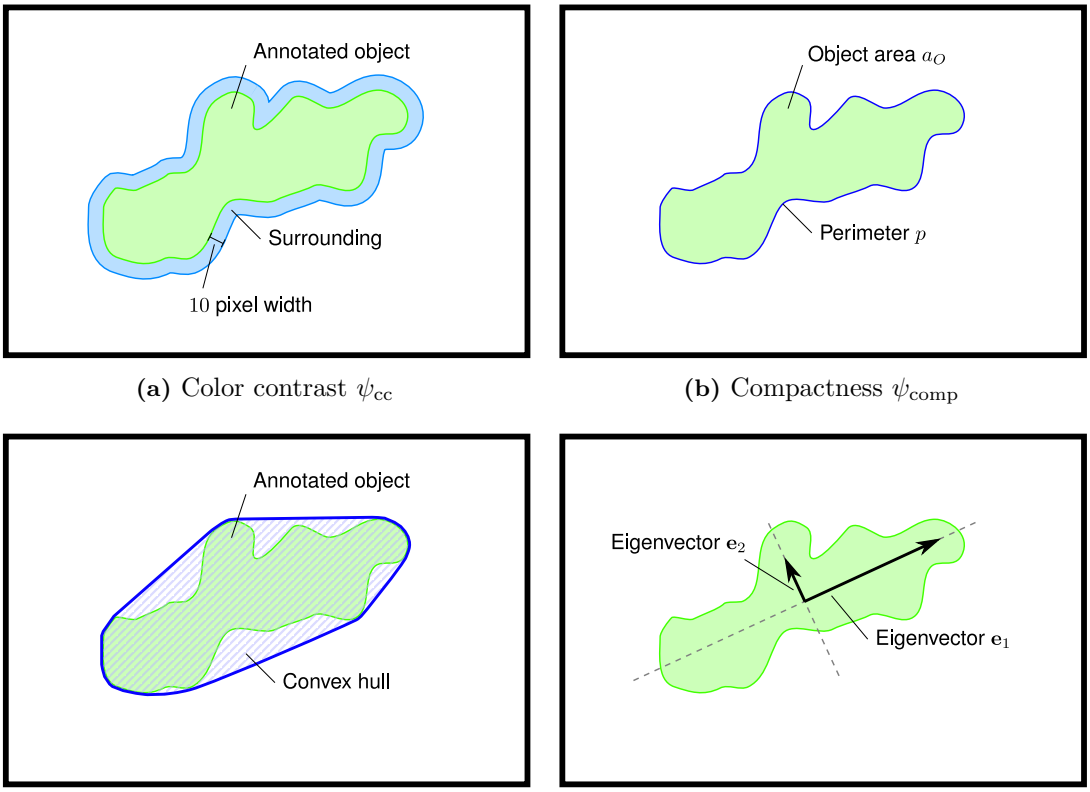
8.1.1 Definition of Properties

We evaluate the object proposal generation results w.r.t. six properties of the annotated objects described below. The six properties are the color contrast, the textureness, three properties covering the object's shape (compactness, convexity, and eccentricity), and the object class.

Color Contrast

An essential property for discovering an object is the *color contrast* of the object w.r.t. its surrounding. A strong color contrast supports discovering objects in humans and monkeys [Rajalingham et al., 2018] as well as computer vision systems [Fan et al., 2020]. Therefore, we measure the color contrast to assess its influence on the object proposal generation results. To calculate the color contrast between the object and its surrounding, we consider two regions and compare their color information. The two regions are the annotated object (green area in Fig. 8.1(a)) and a 10 pixel wide region around the annotated object representing the surrounding (blue area in Fig. 8.1(a)). Note that varying the fixed width of the surrounding region based on the annotated object's size hardly changes the subsequent results.

To calculate the regions' color information, we utilize the LAB colorspace and generate normalized histograms with 16 bins per color channel. Hence, a descriptor of length $3 \cdot 16$ represents each region. We compare the two histogram-based descriptors using symmetric *Kullback-Leibler divergence* to determine the color contrast. Thus, given the histogram-based



(a) Color contrast ψ_{cc} **(b)** Compactness ψ_{comp}

(c) Convexity ψ_{conv} **(d)** Eccentricity ψ_{ecc}

Figure 8.1: Visualizations of the main elements for calculating the properties color contrast ψ_{cc} (a), compactness ψ_{comp} (b), convexity ψ_{conv} (c), and eccentricity ψ_{ecc} (d).

descriptors of the annotated object $h_{\text{obj}}(x_{\text{bin}})$ and the surrounding region $h_{\text{surr}}(x_{\text{bin}})$, we define the color contrast ψ_{cc} as

$$\psi_{\text{cc}} = \sum_{x_{\text{bin}}=1}^{3\cdot 16} h_{\text{obj}}(x_{\text{bin}}) \log\left(\frac{h_{\text{obj}}(x_{\text{bin}})}{h_{\text{surr}}(x_{\text{bin}})}\right) + \sum_{x_{\text{bin}}=1}^{3\cdot 16} h_{\text{surr}}(x_{\text{bin}}) \log\left(\frac{h_{\text{surr}}(x_{\text{bin}})}{h_{\text{obj}}(x_{\text{bin}})}\right). \tag{8.1}$$

Overall, low values for ψ_{cc} correspond to a low color contrast and vice versa.

Textureness

To complement the color contrast, we also measure the *textureness* of objects. This is important since CNN-based systems tend to focus on texture when learning from natural images [Geirhos et al., 2018a; Islam et al., 2021]. We utilize the *Gray Level Co-occurrence Matrix* (GLCM) [Haralick et al., 1973] to capture the textureness of an object. GLCMs are frequently used as texture descriptors in multiple applications [Christodoulou et al., 2003; Schwartz et al., 2009; Peña-Barragán et al., 2011]. The GLCM $\mathbf{A}^{d,\theta}$ is an accumulator of size $l \times l$ for an image with l different gray levels. It accumulates how frequently combinations of gray values occur in an image for pixels at a pre-defined distance d and a pre-defined orientation θ. For instance, if the distance is 1 and the orientation is 0°, we compare each pixel with its lower neighbor. Hence, the two gray values g_1 and g_2 are extracted, and the GLCM entries with these gray values as indices ($\mathbf{A}_{g_1,g_2}$ and $\mathbf{A}_{g_2,g_1}$) are incremented. If an image is textureless, only the entries around the GLCM's main diagonal are nonzero. In

contrast, entries in other parts of the GLCM will be nonzero if the image features textured areas. Note that a GLCM is constructed for one pair of d and θ. Hence multiple GLCMs are necessary to capture textures with different orientations and scales. Based on the GLCM, different values of $\mathbf{A}^{d,\theta}$ are extracted, including the energy or the contrast to aggregate the GLCM to one value [Haralick et al., 1973].

To measure the textureness of an annotated object with a GLCM, we convert the RGB image to gray values and generate a GLCM on the area covered by the annotated object. For the distance d, we use three values adapted to the size of the annotated object. In contrast, the orientation θ is fixed to the values ($0°, 45°, 90°$, and $135°$), independent of the object size. Hence, we generate 12 GLCMs per annotated object. For each GLCM, we determine the contrast $\psi^{d,\theta}_{\text{GLCM contrast}}$ of $\mathbf{A}^{d,\theta}$ to represents the textureness. The contrast is the sum over $\mathbf{A}^{d,\theta}$ weighted by the gray value difference that each entry represents [Haralick et al., 1973]. Thus, the contrast is defined as

$$\psi^{d,\theta}_{\text{GLCM contrast}} = \sum_{g_1=1}^{l} \sum_{g_2=1}^{l} \mathbf{A}^{d,\theta}_{g_1,g_2} (g_1 - g_2)^2. \tag{8.2}$$

Low values for $\psi^{d,\theta}_{\text{GLCM contrast}}$ represent a low textureness and vice versa. Since we have 12 different GLCMs, we accumulate all $\psi^{d,\theta}_{\text{GLCM contrast}}$ results for the different combinations of d and θ by taking the maximum and denoting it as ψ_{tex}. Therefore, we utilize the contrast of the dominant GLCM across the different distances and orientations.

Object Shape

Following the definition of two properties derived from RGB or gray values, we now focus on properties that only capture the object's shape. Therefore, the following properties are all defined based on a binary image representing the annotated object.

Compactness *Compactness* is a shape property that has previously been used for generating object proposals [Yanulevskaya et al., 2014; Werner et al., 2015]. We measure the compactness of an annotated object as circularity following a classic definition [Gonzalez and Woods, 2018] used by Werner et al. [2015] and others. Hence, the compactness ψ_{comp} is defined as

$$\psi_{\text{comp}} = \frac{4\pi a_O}{p^2}. \tag{8.3}$$

It compares the annotated object to a circle by calculating the ratio between the area of the annotated object (a_O) and the annotated object's perimeter (p) as visualized in Fig. 8.1(b). If the annotated object is a circle, $\psi_{\text{comp}} = 1$. Accordingly, the score decreases if the annotated object diverges from the circular shape.

Convexity The second shape property, *convexity*, has not only been used to generate object proposals [Karpathy et al., 2013; Yanulevskaya et al., 2014; Werner et al., 2015] but also supports perceptual grouping in human visual perception [Kanizsa and Gerbino, 1976]. To measure the convexity of an annotated object, we follow a standard definition [Chaki and Dey,

2020][1] used by Werner et al. [2015]. First, we calculate the convex hull H of the annotated object O, visualized in Fig. 8.1(c) by the hatched blue area. Subsequently, convexity is the ratio between the area of the annotated object a_O and the convex hull a_H:

$$\psi_{\text{conv}} = \frac{a_O}{a_H}. \tag{8.4}$$

Similar to compactness, high values for convexity correspond to objects that are convex and vice versa.

Eccentricity As a final shape property, we measure how elongated an annotated object is. This property is calculated as the *eccentricity* [Gonzalez and Woods, 2018], similar to Werner et al. [2015]. To determine an annotated object's eccentricity, we first determine the eigenvalues (λ_1 and λ_2) and eigenvectors ($\mathbf{e}_1$ and $\mathbf{e}_2$) of the covariance matrix calculated from the second-order central moments of the annotated object's coordinates. As a result, the larger eigenvalue λ_1 is associated with the eigenvector $\mathbf{e}_1$ along the major axis of the object. Accordingly, the smaller eigenvalue λ_2 is associated with the eigenvector $\mathbf{e}_2$ along the object's minor axis. This is visualized in Fig. 8.1(d) with the eigenvectors scaled by the eigenvalues. Finally, the ratio of the eigenvalues λ_1 and λ_2 is used to define the eccentricity ψ_{ecc}:

$$\psi_{\text{ecc}} = \sqrt{1 - \frac{\lambda_2}{\lambda_1}}. \tag{8.5}$$

If the eccentricity equals 0, the annotated object's principal axes have the same length, similar to a circle or a square. The more the eccentricity diverges from 0, the more elongated the annotated object is.

Object Class

The final object property that we define is the object's class[2]. For the COCO dataset [Lin et al., 2014] and the LVIS dataset [Gupta et al., 2019], object classes are annotated such that no further calculation or annotation is necessary. As described in Sec. 2.2.2, the COCO dataset features annotated objects of 80 classes, while the LVIS dataset distinguishes 1723 classes with a higher level of detail.

8.1.2 Datasets and Methods

After defining the object properties, we present the datasets and object proposal generation methods utilized in this chapter. As datasets, we use the challenging COCO [Lin et al., 2014] and LVIS [Gupta et al., 2019] test datasets for our evaluation. However, the COCO dataset is only used in the object class-specific evaluation, while all other evaluations are carried out on the LVIS dataset. The LVIS dataset is more suitable for calculating the other object properties since the annotations allow a more precise analysis of the object shapes and more accurately distinguish foreground from background. In contrast, we omit the LVIS dataset for the object

[1] Chaki and Dey [2020] use the term solidity for this measure.

[2] We follow a very abstract concept of objects here and view the object class as an object property for a unified presentation.

Table 8.1: Classification of the evaluated object proposal generation methods w.r.t. to the use of superpixels (second column), the use of a CNN (third column), and the concept followed for integrating the CNN (fourth row). The concepts are the multi-shot concept and the one-shot concept described in Sec. 3.2.3. If no CNN is utilized, neither concept is applied.

Method	Superpixels	CNN	Concept
MCG	✓	✗	-
COB	✓	✗	-
DeepMask	✗	✓	multi-shot
SharpMask	✗	✓	multi-shot
FastMask	✗	✓	one-shot
AttentionMask	✗	✓	one-shot
SAM+FH	✓	✓	one-shot
SAM+EA-ETPS	✓	✓	one-shot
SAM+DeepFH	✓	✓	one-shot

class-specific evaluation due to the large number of classes. Note that despite the differences in annotations, the property-specific results are similar on both datasets.

Given the object properties and the datasets, we compare the results of nine relevant object proposal generation systems. Besides our proposed systems AttentionMask (see Ch. 4), SAM+FH (see Ch. 5), SAM+EA-ETPS (see Ch. 6), and SAM+DeepFH (see Ch. 7), we also evaluate MCG [Arbeláez et al., 2014; Pont-Tuset et al., 2017], COB [Maninis et al., 2016, 2017], DeepMask [Pinheiro et al., 2015], SharpMask [Pinheiro et al., 2016], and FastMask [Hu et al., 2017a]. This allows us to capture the influence of superpixels, CNNs, and the one-shot as well as the multi-shot concept on the results. Table 8.1 presents a classification of the systems w.r.t. those categories.

8.1.3 Evaluation Measures

To assess the performance of the different object proposal generation methods, we use Average Recall (AR) as described in Sec. 2.2.3, similar to the evaluations in Ch. 4 - Ch. 7. AR jointly evaluates how many objects are discovered and how well they are segmented. Similar to the definition of the size-specific ARs in Sec. 2.2.3, we use property-specific ARs for the defined properties. For each property except for the object class, we sort all annotated objects based on the property and split them into eight bins with equal cardinality. Hence, each bin represents a range of property values from low values to high values. For the object class-specific evaluation, we define a bin per object class, leading to 80 bins for the COCO dataset. Subsequently, we report the AR@100 for the annotated objects of each bin generating 80 (object class) or 8 (other properties) property-specific results per object proposal generation method.

In addition to the property-specific ARs, we also calculate the *Pearson correlation coefficient* [Illowsky and Dean, 2018] between different properties. This enables us to discover dependencies between the properties that are hidden in the data. The Pearson correlation

coefficient r_{ψ_p,ψ_q} measures the linear correlation between properties ψ_p and ψ_q across all n annotated objects based on the covariance and the individual variances:

$$r_{\psi_p,\psi_q} = \frac{n\sum_{i=1}^{n}(\psi_{p_i}\cdot\psi_{q_i}) - \sum_{i=1}^{n}\psi_{p_i}\sum_{i=1}^{n}\psi_{q_i}}{\sqrt{\left(n\sum_{i=1}^{n}\psi_{p_i}^2 - \left(\sum_{i=1}^{n}\psi_{p_i}\right)^2\right)\left(n\sum_{i=1}^{n}\psi_{q_i}^2 - \left(\sum_{i=1}^{n}\psi_{q_i}\right)^2\right)}}, \tag{8.6}$$

with ψ_{p_i} denoting the value of the property ψ_p for the ith object. The result r_{ψ_p,ψ_q} is in the interval of $[-1,1]$. While values around 0 indicate no correlation, values closer to 1 or -1 indicate a strong correlation or anti-correlation. For the interpretation of the Pearson correlation coefficient, we use the rule of thumb by Hinkle et al. [2003]: If r_{ψ_p,ψ_q} is in the range of $-0.3\cdots+0.3$, there is no relevant correlation, while $r_{\psi_p,\psi_q} < -0.9$ or $r_{\psi_p,\psi_q} > +0.9$ indicate a strong (anti-) correlation. For the remaining ranges, intermediate interpretations apply [Hinkle et al., 2003]. To generate significant results, we always use all 50,754 annotated objects in the LVIS test dataset to generate r_{ψ_p,ψ_q}.

8.2 Results

Based on the previously described setup, we present the results of our extended evaluation in this section. Similar to the description of the object properties, we discuss the influence of each property individually. Additionally, we first discuss the influence of the object size on the results using the same framework as a baseline for the subsequent analyses.

8.2.1 Size

The previous evaluations, e.g., in Sec. 4.4.1 and Sec. 7.3.2, showcased a strong influence of the annotated object's size on the object proposal generation results. This influence is highlighted by the results in terms of the size-specific AR^S, AR^M, and AR^L measures and is in line with the findings of other authors [Pinheiro et al., 2015, 2016; Hu et al., 2017a]. To better judge the influence of the other object properties described in Sec. 8.1.1, we apply the same evaluation on the size of the annotated objects. Thus, we generate eight size-specific ARs for each object proposal generation system. The results of this baseline experiment are visualized in Fig. 8.2.

The results in Fig. 8.2 show that the performance of all systems degrades for small objects. The average improvement in terms of AR@100 from the bin with the smallest objects to the bin with the largest objects is 0.450 across all systems. These results are in line with the results in the previous evaluations (see Sec. 4.4.1 or Sec. 7.3.2). Moreover, we are now able to show that the smallest 12.5% of the objects in the LVIS test dataset (first bin) are almost not discovered at all. Even the AR@100 for SAM+DeepFH is at a low level (0.003) for these objects. Across most other bins, the results are similar compared to the previous evaluations based on AR^S, AR^M, and AR^L. To explain this general behavior, we refer to Ch. 4.

Overall, the results in Fig. 8.2 support the previous size-specific results and serve as a baseline for judging the influence of the other object properties.

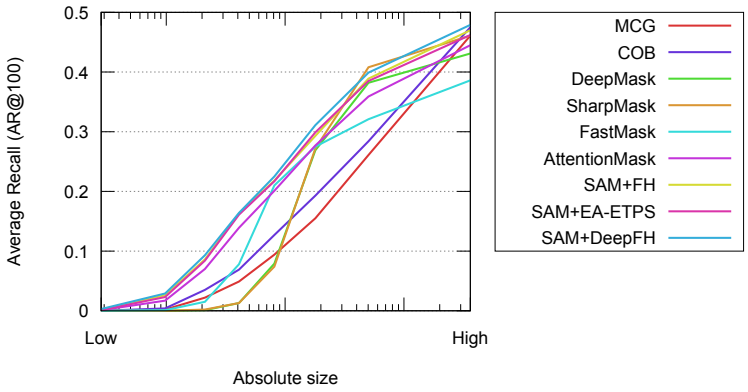


Figure 8.2: Size-specific results in terms of Average Recall (AR@100) on the LVIS test dataset for the nine object proposal generation methods evaluated in this chapter. Note that log scale is applied to the x-axis denoting the annotated object's size, while the y-axis is truncated at 0.5 for improved visibility. The results for SAM+FH and SAM+EA-ETPS are almost at an identical level throughout the plot.

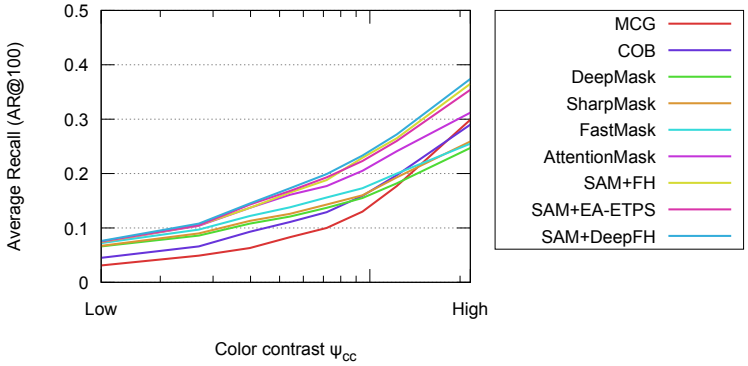


Figure 8.3: Color contrast-specific results in terms of Average Recall (AR@100) on the LVIS test dataset for the nine object proposal generation methods evaluated in this chapter. Note that log scale is applied to the x-axis denoting the annotated object's color contrast, while the y-axis is truncated at 0.5 for improved visibility. The results for SAM+FH and SAM+EA-ETPS are almost at an identical level throughout the plot.

8.2.2 Color Contrast

The first new object property we evaluate is the color contrast ψ_{cc}. The results of the evaluation w.r.t. the color contrast in Fig. 8.3 show that the color contrast influences all evaluated object proposal generation systems. However, this influence is not as strong as the influence of the object size discussed above. On average, the AR@100 improves by 0.237 from the bin with the low-contrast objects to the bin with the high-contrast objects (0.450 for size).

Analyzing the results in Fig. 8.3 in more detail shows that the color contrast does not influence all systems equally. For instance, the improvement in terms of AR@100 between low-contrast objects (first bin) and high-contrast objects (last bin) is 0.268 for MCG and only 0.183 for FastMask. Similarly, the improvement between those two bins is 0.238 for AttentionMask and 0.291 for SAM+FH. Hence, systems utilizing superpixels (MCG and SAM) benefit more from a strong color contrast than pure CNN-based systems. This is due to the construction of

Figure 8.4: Example images from the LVIS test dataset with objects exhibiting low color contrast (upper two rows) and high color contrast (lower two rows). The first column shows the input image, while the second column depicts the best fitting AttentionMask proposal for the annotated object presented in the third column. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) for one annotated object per image is visualized. Input images and annotations taken from the LVIS dataset [Gupta et al., 2019].

most superpixel segmentation methods that rely on color or brightness contrasts. Moreover, it is possible to combine the positive effect of superpixels on high-contrast objects and the stronger performance of CNN-based systems on low-contrast objects, as the results for the SAM-based systems indicate.

To visualize the effect of low or high color contrast, we show qualitative results of AttentionMask on both types of objects in Fig. 8.4. For the low-contrast objects (first and second row), it is visible that AttentionMask has problems distinguishing the annotated object from the similarly colored background. In the case of the sausage, AttentionMask is unable to discover the object. In contrast, the elephant in the second example is discovered, but the elephant's extent is only roughly estimated. For instance, the cub next to the discovered elephant is integrated into the proposal, while the opposite effect is visible on the rightmost leg. The two examples in the lower part show high-contrast objects that AttentionMask easily discovers.

No relevant correlation is evident when comparing the color contrast ψ_{cc} to the other object properties. Hence, color contrast is not related to the other properties including the object size.

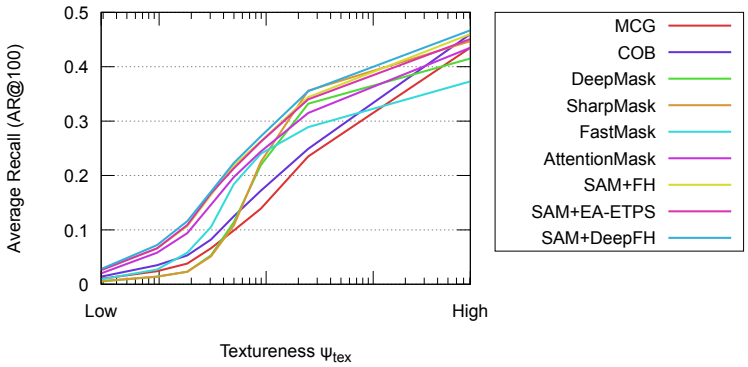


Figure 8.5: Texture-specific results in terms of Average Recall (AR@100) on the LVIS test dataset for the nine object proposal generation methods evaluated in this chapter. Note that log scale is applied to the x-axis denoting the annotated object's textureness, while the y-axis is truncated at 0.5 for improved visibility. The results for SAM+FH and SAM+EA-ETPS are almost at an identical level throughout the plot.

This is also supported by the qualitative results in Fig. 8.4 that include low and high color contrast for small (sausage and kite) and large (elephant and cat) objects.

Overall, the color contrast between the object and the surrounding substantially influences the object proposal results. Additionally, superpixel-based object proposal generation methods like MCG or SAM benefit more from a strong color contrast. The results also show that it is possible to combine the positive effects of superpixels on high-contrast objects and of CNN-based methods on low-contrast objects.

8.2.3 Textureness

Inspired by findings that show a strong focus of CNNs on texture [Geirhos et al., 2018a; Islam et al., 2021], we investigate the influence of the object's textureness ψ_{tex} on the results of object proposal generation methods. The results in terms of the texture-specific ARs in Fig. 8.5 indicate a strong influence of the objects' textureness. The improvement in terms of AR@100 from the low-textureness bin to the high-textureness bin is 0.421 across all methods. This is similar to the influence of the size discussed previously (0.421 vs. 0.450). One explanation for the strong influence of textureness on all CNN-based methods is the focus of such methods on texture when learning from data as mentioned above. Hence, if only a weak texture exists, the CNN-based systems will struggle to discover the objects.

This interpretation is also supported by the qualitative results in Fig. 8.6 that visualize annotated objects with low and high textureness as well as the proposals generated by AttentionMask. The wristband in the first example or the sheep in the background in the second example have virtually no recognizable texture on the image plane. Hence, AttentionMask misses both objects. In contrast, the sheep in the third example has a rich uniform texture originating from its wool. Similarly, the bus in the final example features various textures due to the windows, the wheel cap, and the advertisements. Hence, both objects have strong textures that allow AttentionMask to discover them.

Besides the focus of CNNs on texture, another interpretation opens up when comparing the size-specific results (see Fig. 8.2) and the texture-specific results (see Fig. 8.5) in detail. Both

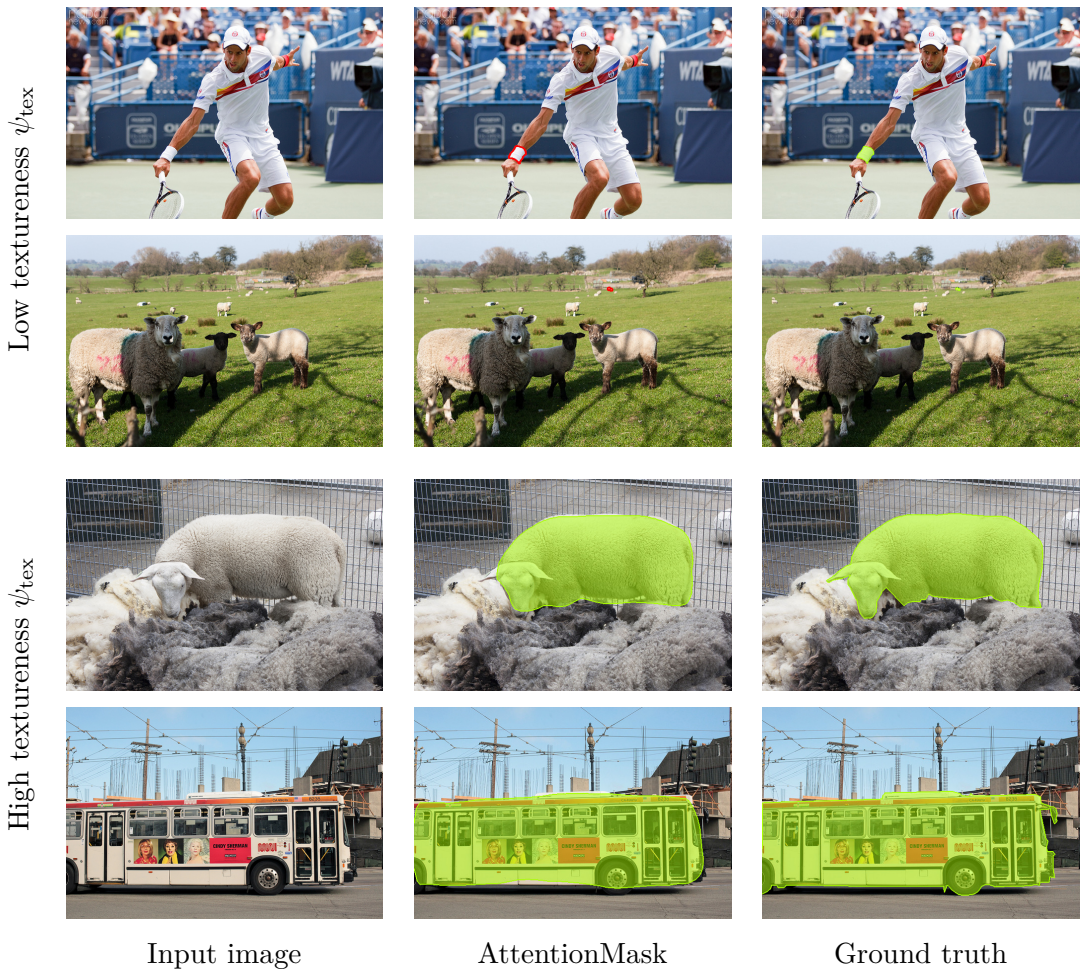


Figure 8.6: Example images from the LVIS test dataset with non-textured objects (upper two rows) and textured objects (lower two rows). The first column shows the input image, while the second column depicts the best fitting AttentionMask proposal for the annotated object presented in the third column. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) for one annotated object per image is visualized. Input images and annotations taken from the LVIS dataset [Gupta et al., 2019].

results are similar for most methods. Accordingly, the Pearson correlation coefficient between the object size and textureness is 0.855 across all annotated objects. Hence, both properties are strongly correlated. The strong correlation is the result of a lack of texture in small objects due to the low resolution on the image plane. Even if an object originally has a rich texture, this texture vanishes, as the object is projected onto the image plane since the spatial resolution of the image is too coarse to capture the texture. This effect is visible from the examples in Fig. 8.6. The texture of the sheep in the background in the second example is not visible on the image plane due to the sheep's size. In contrast, the large sheep in the third row has a rich uniform texture. Thus, the size of an object and the object's texture on the image plane are closely related.

Overall, we derive a strong influence of textureness on object proposal generation results. However, textureness and size are closely related in terms of correlation and causality. Given that several systems are technically unable to learn small or tiny objects (e.g., see training procedure of FastMask in Sec. 3.2.3), we assume that the influence of the size is stronger than

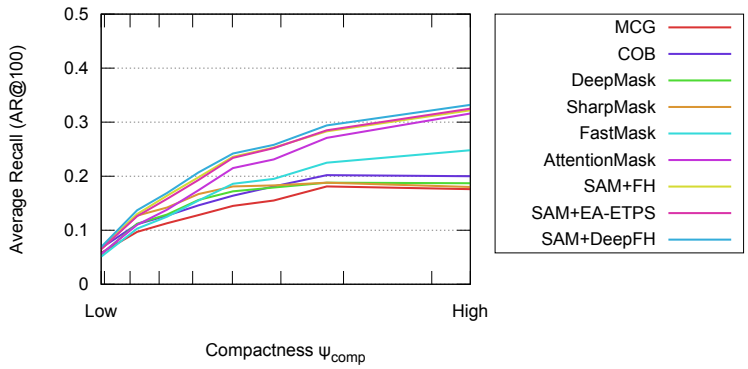


Figure 8.7: Compactness-specific results in terms of Average Recall (AR@100) on the LVIS test dataset for the nine object proposal generation methods evaluated in this chapter. Note that log scale is applied to the x-axis denoting the annotated object's compactness, while the y-axis is truncated at 0.5 for improved visibility. The results for SAM+FH and SAM+EA-ETPS are almost at an identical level throughout the plot.

the influence of textureness. Moreover, the results of MCG follow the general trend of the CNN-based methods, although MCG does not utilize CNNs. Hence, the effect of the size is likely to be more dominant.

8.2.4 Object Shape

After discussing the influence of color and texture, we investigate the shape properties of the annotated objects in the subsequent sections. Note that the shape properties only rely on the binary segmentation mask of the annotated object and do not consider color or texture at all.

Compactness and Convexity

First, we jointly discuss the influence of compactness (ψ_{comp}) and convexity (ψ_{conv}) on the object proposal generation results since both properties measure similar yet slightly different qualities. While compactness compares the object to a circle, convexity matches the object to its convex hull. However, both properties prefer objects without significant notches in their shape. The compactness-specific ARs in Fig. 8.7 and the convexity-specific ARs in Fig. 8.8 indicate a substantial influence of both properties on the overall results. Highly compact or highly convex objects are generally easier to discover and vice versa. The strength of the influence is similar to color contrast. For compactness, the results in terms of AR@100 increase on average by 0.184 from the lowest bin to the highest bin, while the increase for convexity is 0.191 on average.

A closer look at both results reveals that the influence of compactness and convexity is strongest on one-shot object proposal generation systems like AttentionMask. This results from the strong downsampling in the one-shot concept discussed in Ch. 5. Therefore, the segmentations are generated on a low resolution and subsequently upsampled through bilinear interpolation. The interpolation leads to blob-like proposals as visible from the results of AttentionMask in Fig. 8.9. For instance, AttentionMask only discovers the main part of the kite in the first row with a blob-like proposal while omitting details like the tails. Similarly,

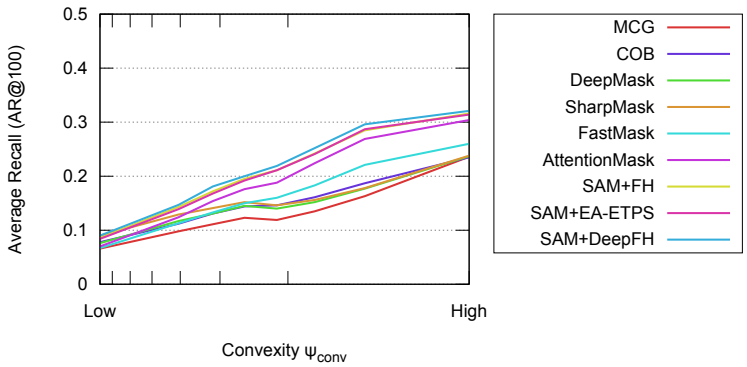


Figure 8.8: Convexity-specific results in terms of Average Recall (AR@100) on the LVIS test dataset for the nine object proposal methods evaluated in this chapter. Note that log scale is applied to the x-axis denoting the annotated object's convexity, while the y-axis is truncated at 0.5 for improved visibility. The results for SAM+FH and SAM+EA-ETPS are almost at an identical level throughout the plot.

AttentionMask misses details of the ships' shape in the second row, like the antennas, the bow, and the stern. These imprecise proposals lead to a more substantial drop in AR@100 on such objects for one-shot approaches. In contrast, the results in the third and fourth row in Fig. 8.9 indicate that the blob-like proposals of AttentionMask match convex or compact objects like the orange or the stop sign very well.

The results in Fig. 8.7 and Fig. 8.8 revealed similarities between compactness and convexity. These similarities are supported by the Pearson correlation coefficient $r_{\psi_{\text{comp}},\psi_{\text{conv}}}$ that indicates a strong correlation ($r_{\psi_{\text{comp}},\psi_{\text{conv}}} = 0.806$) between compactness and convexity. This is plausible since compactness and convexity assess similar properties as discussed above. Additionally, perfect compactness (circle shape) will automatically lead to perfect convexity. In contrast, high convexity does not automatically lead to high compactness as the baseball bat in the final row of Fig. 8.9 shows. Besides the correlation between compactness and convexity, only a medium correlation between compactness and eccentricity was found ($r_{\psi_{\text{comp}},\psi_{\text{ecc}}} = 0.589$).

Overall, the results showcase a substantial influence of compactness and convexity on the object proposal generation results. Moreover, the properties are closely related in terms of correlation, which is in line with the properties' definitions. The results also revealed that the influence of both properties is stronger in one-shot object proposal generation systems. We attributed this to the stronger downsampling in these systems and the subsequent interpolation-based upsampling of the object proposals.

Eccentricity

The final shape property that we investigate is the eccentricity of an annotated object (ψ_{ecc}). Figure 8.10 presents the results for the eccentricity-specific ARs on the nine object proposal generation methods. These results show an influence of similar strength compared to color contrast, compactness, and convexity. However, the influence is inverted. Hence, low-eccentricity objects, which do not have an elongated shape, are easier to discover than high-eccentricity objects. The improvement from high-eccentricity objects to low-eccentricity objects is 0.215 in terms of AR@100.

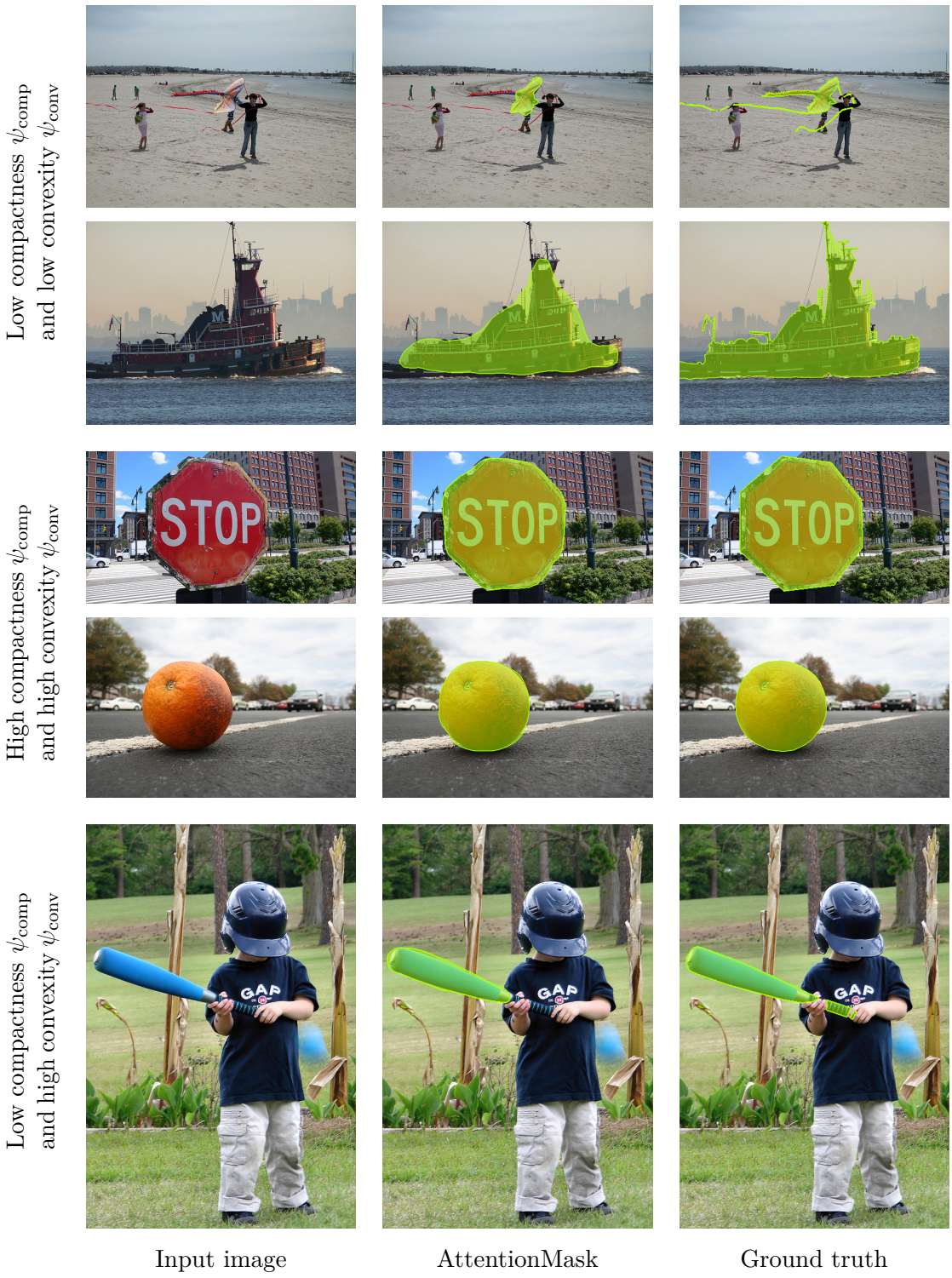


Figure 8.9: Example images from the LVIS test dataset with non-compact and non-convex objects (first two rows), compact and convex objects (third and fourth row) as well as non-compact but convex objects (final row). The first column shows the input image, while the second column depicts the best fitting AttentionMask proposal for the annotated object presented in the third column. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) for one annotated object per image is visualized. Input images and annotations taken from the LVIS dataset [Gupta et al., 2019].

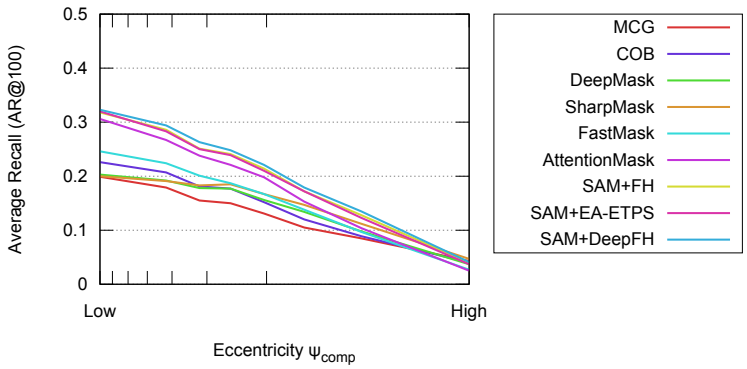


Figure 8.10: Eccentricity-specific results in terms of Average Recall (AR@100) on the LVIS test dataset for the nine object proposal generation methods evaluated in this chapter. Note that log scale is applied to the x-axis denoting the annotated object's eccentricity, while the y-axis is truncated at 0.5 for improved visibility. The results for SAM+FH and SAM+EA-ETPS are almost at an identical level throughout the plot.

Investigating the results further reveals that the performance of AttentionMask and FastMask drops significantly with rising eccentricity. For the bin with the second-highest eccentricity, both methods drop below SharpMask while dropping below all other methods on high-eccentricity objects (last bin). Hence, AttentionMask and FastMask suffer most from elongated objects. This effect results from of the systems' design, extracting square windows (10×10) from the feature pyramid. Hence, the windows do not adapt to elongated objects. In contrast, SharpMask and DeepMask use windows with multiple aspect ratios extracted from the image pyramid, while MCG, COB, and the variations of SAM utilize superpixels to circumvent (MCG and COB) or mitigate (SAM) the effect.

The influence of the eccentricity on AttentionMask is also visible in the qualitative results in Fig. 8.11. The first two examples in Fig. 8.11 show typical low-eccentricity objects. Both objects are discovered by AttentionMask since they fit into square windows. In contrast, the objects in the final two examples do not fit into square windows due to the strong eccentricity. As a result, AttentionMask misses the skis and only captures the upper part of the horse. Missing the legs of the horse can result from two scenarios. First, the legs were not part of the square window extracted from the feature pyramid. Second, the legs were included, but the window was extracted from a pyramid layer that is supposed to discover larger objects. Hence, the legs are too thin to get discovered. Note that both reasons are tied to the high eccentricity of the object.

As previously discussed and visible from the first example in Fig. 8.11, eccentricity and compactness are correlated ($r_{\psi_{\text{comp}},\psi_{\text{ecc}}} = 0.589$). This correlation is apparent since several objects with a low eccentricity have a circle-like shape, leading to high compactness. However, as visible from the bear in the second example in Fig. 8.11, this is not the case for all low-eccentric objects. No relevant correlation to any other discussed property was found for eccentricity.

In general, the results w.r.t. the eccentricity of the objects indicate a substantial influence, especially on AttentionMask and FastMask that discover objects based on square windows. In contrast, the results of the superpixel-based approaches (e.g., SAM) that are not limited by any windows and suffer less from elongated objects.

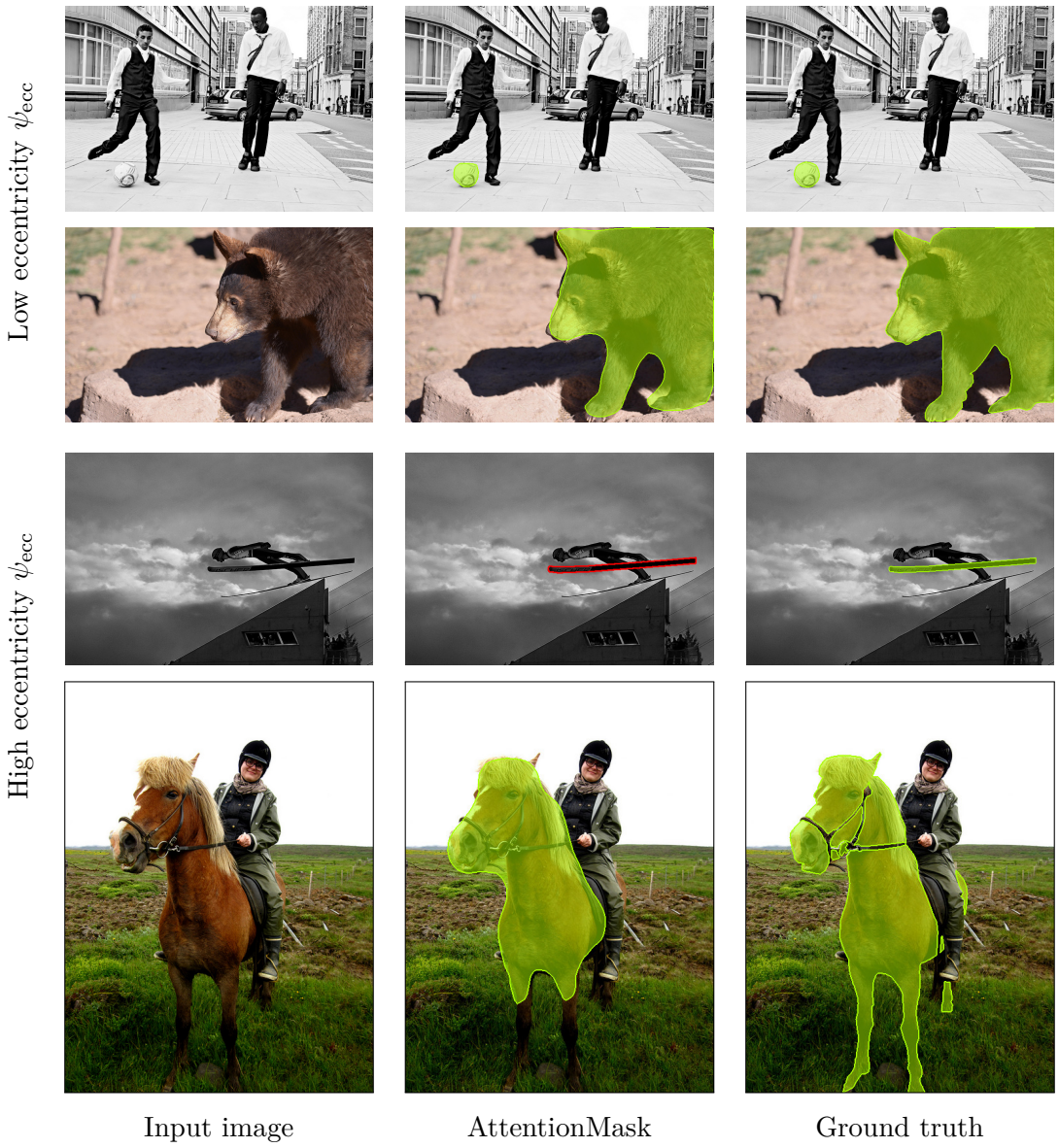


Figure 8.11: Example images from the LVIS test dataset with low-eccentricity objects (upper two rows) and high-eccentricity objects (lower two rows). The first column shows the input image, while the second column depicts the best fitting AttentionMask proposal for the annotated object presented in the third column. The black frames around the images in the final row are only added for visualization and are not part of the image. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) for one annotated object per image is visualized. Input images and annotations taken from the LVIS dataset [Gupta et al., 2019].

Table 8.2: Class-specific results in terms of Average Recall for 100 proposals (AR@100) as well as class-specific properties on the COCO test dataset. The AR@100 is the mean across the evaluated object proposal generation methods on annotated objects of a specific class. Similarly, the properties ψ_{cc}, ψ_{tex}, ψ_{conv}, ψ_{ecc}, and the object size are averaged across the annotated objects of the respective class. The final column denotes the number of annotations per class across the COCO training dataset. The upper part of the table includes difficult object classes, while the central part shows results on simple object classes. For comparison, we give the average across the 80 classes in the final part of the table.

Class	AR@100↑	Size	ψ_{cc}	ψ_{tex}	ψ_{conv}	ψ_{ecc}	No. of annotations
Ski	0.068	1,958	0.364	127.8	0.669	0.961	4,684
Fork	0.082	2,155	0.510	151.2	0.617	0.953	3,914
Sports ball	0.175	879	0.949	32.1	0.940	0.516	4,361
Bear	0.518	54,701	0.904	2,093.4	0.823	0.767	903
Elephant	0.537	28,832	0.601	1,470.7	0.753	0.746	3,880
Bus	0.559	44,390	0.385	2,222.3	0.899	0.730	4,321
Average	0.323	15,852	0.671	736.0	0.822	0.812	7,473

8.2.5 Object Class

Switching to high-level information, we investigate the influence of the annotated object class on the class-agnostic object proposal generation results. Unlike the previous analyses, we conduct this evaluation on the COCO test dataset with only 80 object classes. The class-specific AR@100 for the 80 object classes in the COCO dataset varies between 0.068 (*ski*) and 0.609 (*toilet*) when averaged across the nine object proposal generation methods. Overall, the per-class average in terms of AR@100 is 0.323 with a standard deviation of 0.127. The different object proposal generation methods show similar patterns on different AR-levels across the results on the different object classes. Therefore, we average the results of the different methods for our subsequent analysis of the object classes' influence.

After giving an overall impression, we will discuss the results of six object classes in more detail. The first part of Tab. 8.2 shows the results for the three difficult classes *ski*, *fork*, and *sports ball*. The difficulty is well documented by the low mean ARs across all proposal methods, which are well below the average across all classes (see bottom row). For the three classes and most other difficult classes, the low AR is related to the small size of the annotated objects, which is in line with earlier findings. The average size of the annotated objects across the three classes is well below 2,500 pixels, while the average across all classes is 15,852. The small size is also visible from the results in Fig. 8.12, depicting objects of the three classes in typical scenarios. Moreover, the color contrast for the object classes *ski* and *fork* is low, while the eccentricity is high (see first four rows in Fig. 8.12). Since both properties negatively impact the results, the low AR for those classes is easily explained by a combination of multiple factors. In contrast, objects of the class *sports ball* are convex, have a low eccentricity, and a high color contrast. Nevertheless, the results are subpar, since the average size is very small, which is visible from the examples in Fig. 8.12 (final two rows). Therefore, most object proposal generation systems are technically unable to discover those objects.

The second part of Tab. 8.2 presents the results for the three simple object classes *bear*, *elephant* and *bus*. A significant factor for the good results is again the object size that is well

Figure 8.12: Example images from the COCO test dataset with objects from the difficult object classes *ski* (first two examples), *fork* (third and fourth example), and *sports ball* (final two examples). The first column shows the input image, while the second column depicts the best fitting AttentionMask proposals for the annotated objects presented in the third column. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) per object and only the annotated objects from the respective classes are visualized. Input images and annotations taken from the COCO dataset [Lin et al., 2014].

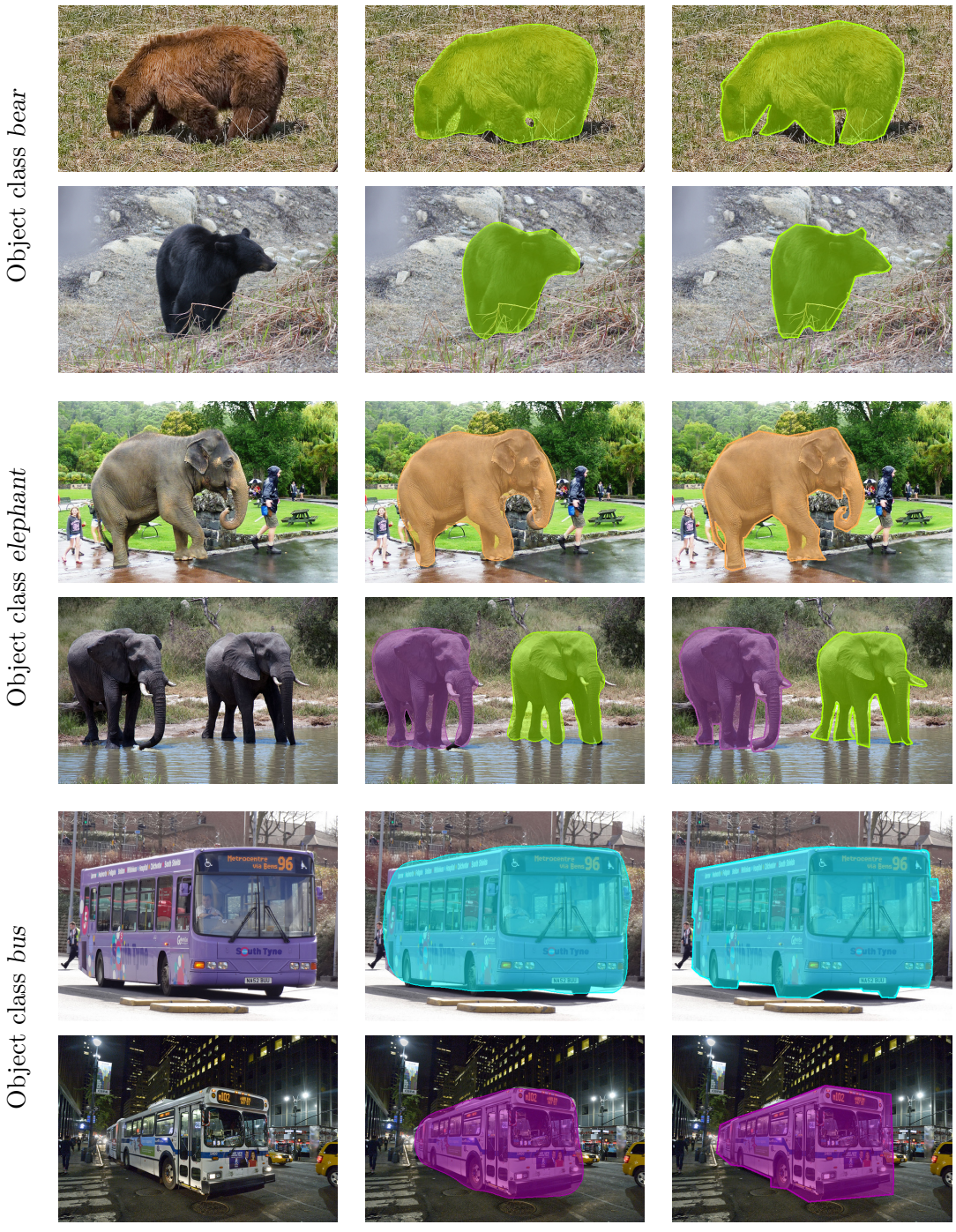


Figure 8.13: Example images from the COCO test dataset with objects from the simple object classes *bear* (first two examples), *elephant* (third and fourth example), and *bus* (final two examples). The first column shows the input image, while the second column depicts the best fitting AttentionMask proposals for the annotated objects presented in the third column. Filled colored contours denote discovered objects, while not filled red contours denote missed objects. Note that only the best fitting proposal (highest IoU) per object and only the annotated objects from the respective classes are visualized. Input images and annotations taken from the COCO dataset [Lin et al., 2014].

above average for the three classes. For instance, objects of the class *bus* are three times larger than the average object. The above-average size is also visible from the qualitative results of the three classes in Fig. 8.13. Most of the images are dominated by an instance of one of the three classes, like the elephant in the third row or the bus in the fifth row. Moreover, all classes have a below-average eccentricity, leading to better results as discussed in Sec. 8.2.4. The objects of the class *bear* also feature a strong color contrast since the bears are primarily black or brown, while the background is usually greenish (see first two rows in Fig. 8.13). Hence, the good results on classes like *bear*, *elephant*, and *bus* are related to size, color contrast, and eccentricity.

Another key finding from Tab. 8.2 is that the number of annotations per object class in the training set does not substantially influence the results. All classes presented in Tab. 8.2 were underrepresented in the training set. For instance, the object class *bear* is only represented by 903 annotated objects across the training set (average: 7,473). Nevertheless, the result in terms of AR@100 is among the best across all classes. In contrast, the class *person* is overrepresented in the training dataset with 181,682 annotated objects. However, the result for the class *person* is below-average (0.249 vs. 0.323), which implies that the systems generalize well across the object classes.

Overall, the results on the 80 object classes in the COCO dataset vary. The differences are similar across the object proposal generation systems and rooted in the properties described previously, like size, color contrast, and eccentricity. Moreover, we showed that the number of training samples per class has no major influence on the results.

8.3 Discussion

In this chapter, we presented our extended evaluation of nine object proposal generation methods on the complex COCO and LVIS datasets. Inspired by the typical size-specific evaluation, we extended the regular evaluation of object proposal generation methods by investigating the influence of six object properties covering color contrast, texture, shape, and the object class on the object proposal generation results. Additionally, we examined the relations between the six properties and the relation to the object size. This detailed analysis extends our previous evaluations and allows a better understanding of the current state of object proposal generation.

The results of our evaluation confirm a strong influence of the object size on the results. Specifically, the smallest 12.5% of the objects in the challenging LVIS test dataset are almost not discovered at all. A similar influence is visible for the textureness of objects, which is not surprising, since both properties are strongly related. Moreover, color contrast, compactness, convexity, and eccentricity substantially influence the results. For the evaluated systems, most of the observed effects are explained by the different characteristics of the systems, like the inherent downsampling process in CNNs or the contrast-based generation of superpixels. Additionally, the results show that our combination of CNNs and superpixels in SAM accumulates the benefits of both processing styles for most properties. Finally, our evaluation revealed that the influence of the different object classes is generally a combination of the previously discussed properties. Moreover, the number of annotated objects per class in the training dataset does not substantially influence the results.

Based on the extended evaluation, we formulate four novel challenges in object proposal generation that are related to methodological limitations of current approaches:

Tiny objects The discovery of tiny objects, which are below approximately 100 pixels in size, is a significant challenge since all systems almost completely ignore those objects. This is related to the strong downsampling in CNN-based methods or insufficient segmentations in superpixel-based methods.

Elongated objects The eccentricity of objects is another major factor for the performance of object proposal generation approaches. Systems utilizing windows or even square windows to roughly discover objects are impacted more than superpixel-based systems.

Complex object shapes The precise segmentation of complex object boundaries with many details poses another challenge. As discussed in Ch. 5, the main reasons are the inherent downsampling process in CNNs or insufficient segmentations, similar to the problem of tiny objects discovery.

Low-contrast objects Discovering objects that have a low color contrast w.r.t. their surrounding is another challenge for object proposal generation systems. This is similar to standard object detection methods as both lack dedicated modules [Fan et al., 2020; Zhai et al., 2021] for such scenarios.

Overall, the evaluation presented in this chapter revealed several object properties that negatively influence the results of object proposal generation methods. These properties were combined into four novel challenges in object proposal generation that point to new potential research directions as we will discuss in Sec. 10.3.

Chapter 9

Applications of AttentionMask

Table of Contents

This chapter presents three applications of our efficient object proposal generation system AttentionMask introduced in Ch 4. As discussed in the introduction, object detection is the most common application for object proposals. According to Hosang et al. [2015], evaluating object proposal generation systems based on Average Recall (AR) is a good surrogate to assess the quality of subsequent object detection systems. Hence, we do not investigate the application of our object proposal generation systems to standard object detection systems.

Alternatively, we focus on three challenging, relevant real-world tasks to showcase AttentionMask's flexibility and robustness. First, in Sec. 9.1, we combine AttentionMask with a tailored classifier for airline logo detection based on two bachelor theses [Heid, 2018; Sadeghi, 2019] and a cooperation with zeroG GmbH, which cumulated in a joint publication [Wilms et al., 2020]. In our setting, this task challenges the robustness of AttentionMask due to adverse weather conditions that significantly degrade the image quality. Additionally, we apply AttentionMask to the challenging medical instrument segmentation in Sec. 9.2 based on a bachelor thesis [Gerlach, 2021] and our joint publication [Wilms et al., 2022a]. In this task, medical instruments have to be localized and segmented in images acquired during

known and unknown types of minimally invasive surgeries. The images are degraded by blood, smoke, and poor illumination, posing again challenges to AttentionMask's robustness. Finally, in Sec. 9.3, we present two AttentionMask variations for apple localization in a complex orchard environment based on the bachelor thesis by Johanson [2021], which led to a joint publication [Wilms et al., 2022b]. This application is challenging, since the apples are small or tiny, and the images are substantially cluttered.

Overall, the three different applications pose significant challenges for AttentionMask in the areas of flexibility, robustness, and the discovery of small and tiny objects. Additionally, all three applications are active research areas involving complex real-world data.

9.1 Airline Logo Detection

The localization and detection of logos, known as logo detection, is an object detection subtask. Logo detection facilitates multiple applications like verifying ad visibility in social media [Gao et al., 2014] or TV broadcasts [Tüzkö et al., 2018]. In recent years, several systems for logo detection [Iandola et al., 2015; Bianco et al., 2017; Fehérvári and Appalaraju, 2019] and a multitude of datasets [Romberg et al., 2011; Fehérvári and Appalaraju, 2019; Tüzkö et al., 2018] were presented.

We argue that logo detection suffers from two drawbacks despite the strong research interest. First, logo detection is mainly addressed by applying application-agnostic object detection systems like Faster R-CNN [Ren et al., 2016] with minor modifications [Iandola et al., 2015; Eggert et al., 2017; Su et al., 2017b]. However, logos are substantially different from general objects. For instance, the color, shape, and pose variety is limited compared to typical object classes like cats or dogs. Specifically, the color and the shape of the logos are usually identical across one class. Since less intra-class variability exists, smaller networks with fewer parameters are sufficient and increase the computational efficiency.

Second, most large-scale logo detection datasets contain clean images or perfect product images [Hoi et al., 2015; Bianco et al., 2017; Fehérvári and Appalaraju, 2019]. While this is sufficient for tasks in controlled indoor environments, it strongly impedes the applicability in outdoor environments. This is visible from Fig. 9.1 that shows logo detection results on a clean image (right) and an image degraded by adverse weather effects (left). On the clean image, the logo on the airplane tail is easily visible and detected by all four systems. In contrast, adverse weather effects like rain or fog make the detection more challenging, resulting in missed or misclassified logos.

To investigate these two drawbacks in more detail, we turn to the detection of airline logos on airplane tails as visualized in Fig. 9.1. This logo detection task is well-suited for this analysis since many clean, high-quality images are available from online databases fueled by planespotters[1]. Adding images showing the effects of adverse weather conditions is also possible since airplanes are usually captured outdoors. Moreover, airline logos on airplane tails are a good example of typical logos, as deformations of logos on the airplane tails are impossible.

[1]For instance, the online database Planespotters.net hosts almost 900,000 high-quality images of airplanes.

(a) Faster R-CNN

(b) YOLOv3

(c) Proposed system without data augmentation

(d) Proposed system with data augmentation

Figure 9.1: Comparison between the application-agnostic object detection systems Faster R-CNN [Ren et al., 2016] (a) and YOLOv3 [Redmon and Farhadi, 2018] (b) as well as our proposed airline logo detection system without (c) and with our data augmentation (d). The airline logo on the right is easily detected by all systems (green frames). In contrast, the airline logos on the left are degraded by rain and fog, which leads to missed or misclassified logos (red or missing frames) except for our system with the proposed data augmentation. Input images taken from Heid [2018] (left image) and Lin et al. [2014] (right image).

In this section, we propose a novel airline logo dataset, a tailored two-stage airline logo detection system based on AttentionMask, and a learning-free data augmentation strategy for adverse weather conditions to investigate the two drawbacks. First, in Sec. 9.1.1, we review related work on the three proposed components: logo detection datasets, logo detection systems, and handling adverse weather conditions in general object detection. Subsequently, we introduce our novel airline logo dataset covering 7038 logos across 41 classes in Sec. 9.1.2. Besides clean images, the dataset also contains challenging images degraded by adverse weather conditions to test the robustness of logo detection systems. Section 9.1.3 presents our dedicated two-stage airline logo detection system based on a variation of AttentionMask (see Ch. 4) and a lightweight, tailored classifier. Moreover, we present a novel, learning-free data augmentation strategy in Sec. 9.1.4 that imitates the effects of adverse weather conditions to improve the results in the absence of degraded training data. Section 9.1.5 evaluates our tailored airline logo detection system on the new dataset and compares the results to application-agnostic object detection systems used in logo detection. This evaluation showcases the merit of a tailored logo detection system and the severe effect of adverse weather conditions on the logo detection results. Subsequently, we demonstrate the positive impact of our learning-free data augmentation strategy for the detection on degraded images. Finally, we discuss the main findings and limitations in Sec. 9.1.6.

9.1.1 Related Work

This section briefly discusses relevant literature on logo detection datasets and logo detection systems. Moreover, we discuss recent advances in general object detection under adverse weather conditions.

Logo Detection Datasets

Datasets for logo detection started with a limited amount of images and classes due to the substantial labeling effort. Proposing a logo detection dataset with bounding box annotations, Kalantidis et al. [2011] collected 40 natural images for 27 logo classes each. Similarly, Romberg et al. [2011] proposed a dataset covering 32 classes with 70 images per class. Driven by the success of CNNs, larger datasets were proposed by Bianco et al. [2017] (7830 images), Hoi et al. [2015] (more than 70,000 images), and Fehérvári and Appalaraju [2019] (almost 300,000 images). However, those datasets focus on clean product images with low complexity and a low amount of clutter. Moving away from clean images, Su et al. [2017a] collect a dataset of product images and more complex natural scenes. However, Su et al. [2017a] provide only image-level annotations. Taking one step further, Tüzkö et al. [2018] propose a dataset (11,504 images) containing only natural scenes with complex settings.

Despite advances toward datasets with logos captured in more complex scenarios, logo detection datasets are dominated by simple, clean product images. Hence, complex scenarios, e.g., including adverse weather conditions, which frequently occur in real-world applications, are not well represented.

Logo Detection

Since the advent of deep learning, logo detection systems moved from hand-crafted features [Romberg et al., 2011; Kalantidis et al., 2011] to CNNs. As discussed earlier, most CNN-based logo detection systems are based on application-agnostic object detection methods like Faster R-CNN [Ren et al., 2016]. Iandola et al. [2015] discover logos with Faster R-CNN and classify them with a CNN-based classifier. Eggert et al. [2017] adapt Faster R-CNN's anchor boxes, detection heads, and backbone to improve results on small logos. Augmenting the training data with synthetic images, Su et al. [2017b] employ Faster R-CNN for improved logo detection results. Fehérvári and Appalaraju [2019] apply the application-agnostic object detectors SSD and YOLO to discover logos. In contrast to those purely CNN-based approaches, Bianco et al. [2017] apply Selective Search [Uijlings et al., 2013] to generate logo proposals and classify them with a small CNN-based classifier.

Overall, CNN-based logo detection systems utilize either application-agnostic object detectors (Faster R-CNN, SSD, and YOLO) or outdated components (Selective Search) to detect logos. Hence, to the best of our knowledge, modern systems specifically tailored to detect logos are lacking.

Adverse Weather Conditions in Object Detection

The effects of adverse weather conditions like fog or rain on object detection systems gained more and more attention in recent years. Several methods tackle the problem in a deep learning-based domain adaptation framework to support object detection systems. Since the methods rely on deep learning, they demand target domain data for training. For instance, Chen et al. [2018] learn a domain classifier with Faster R-CNN to generate domain invariant features, while Hsu et al. [2020] propose a synthetic intermediate domain to allow gradual adaptation. Addressing the difference between global and local domain adaptation, Zhu et al. [2019] and Saito et al. [2019] focus on the alignment of features in object regions. Different from that, RoyChowdhury et al. [2019] generate pseudo-labels for unlabeled target domain video data utilizing the results of an object detector and a temporal consistency constraint. In a different line of research, Halder et al. [2019] propose a physics-based rendering framework to augment source domain images with artificial rain streaks.

In general, most approaches on domain adaptation for adverse weather effects need target domain data for training. However, this data might not be available or only sparsely included in datasets.

9.1.2 PSLogos Dataset

As previously discussed, existing datasets for logo detection lack challenging image settings with adverse weather conditions and other degradations. To overcome this limitation and assess the quality of detectors on degraded images, we propose the novel *PlaneSpottersLogos* (PSLogos) dataset. The PSLogos dataset contains 6563 images of airplanes with bounding box annotations for 7038 airline logos on airplane tails. Most images in the dataset were collected from the online database Planespotters.net[2] and do not suffer from adverse weather conditions. To complement these simple, clean images, we add more challenging images taken from the visitor gallery at Hamburg Airport. These images suffer from adverse weather conditions like rain or fog. Figure 9.2 depicts examples of airline logos on clean and degraded images like in the PSLogos dataset to highlight the difference. The mixture of both image types makes the PSLogos dataset a good foundation for investigating the aforementioned drawbacks in logo detection.

As discussed above, the images in our PSLogos dataset originate from two sources. First it comprises 6311 images captured at various airports worldwide downloaded from Planespotters.net. These images are mostly clean and do not suffer from adverse weather effects. Second, the dataset contains 252 images captured at Hamburg Airport that suffer from adverse weather conditions. The images from Planespotters.net images are between 640×480 and 1600×1216 in size, while the other images are all of the size 3968×2976. We manually annotated the airline logos of 41 airlines on the airplane tails with bounding boxes. The annotations follow a non-uniform distribution with 146 to 186 annotated logos per class.

We split the dataset into four distinct sets as outlined in Tab. 9.1. First, we split the images based on the data source. The first three sets *PS-TRAIN*, *PS-VAL*, and *PS-TEST* consist of the 6311 clean images from Planespotters.net. While PS-TRAIN is used for training, PS-VAL is the validation set, and PS-TEST serves as the first test set of our dataset. Table 9.1

[2]Planespotters.net: `https://www.planespotters.net`

(a) Clean images

(b) Degraded images

Figure 9.2: Examples of airline logos on airplanes in clean images (a) and images degraded by multiple adverse weather effects (b). The green boxes highlight the airline logos. Base images taken from Lin et al. [2014] (a) and Heid [2018] (b)

Table 9.1: Overview of the proposed split of our PSLogos dataset with the number of images, annotations, and classes per set. The final column denotes if the set is used for training, validation, or testing.

Set	No. of images	No. of annotations	No. of classes	Purpose
PS-TRAIN	4396	4517	41	Training
PS-VAL	625	665	41	Validation
PS-TEST	1290	1397	41	Testing
PS-HAM	252	459	13	Testing
Overall	6563	7038	41	

presents the number of images and the number of annotated airline logos per set. Besides these three sets with clean images, all 252 images captured at Hamburg Airport comprise the *PS-HAM* set. This set is exclusively used for testing. As discussed above and visible in Fig. 9.2(b), most images in this set are degraded by adverse weather effects. The most common degradations are due to rainy or foggy conditions. While fog leads to a brightened image, rainy weather induces three main effects. First, the images may be darker due to heavy clouds. Second, the rain streaks reduce the contrast. Lastly, as visible in Fig. 9.2(b), the rain leads to raindrops on the windows of the gallery that distort the image content locally. Hence, these images present significant challenges to the robustness of logo detection methods. Since the availability of airlines is limited at Hamburg Airport, only 13 airline classes are covered across the 459 annotations (see Tab. 9.1).

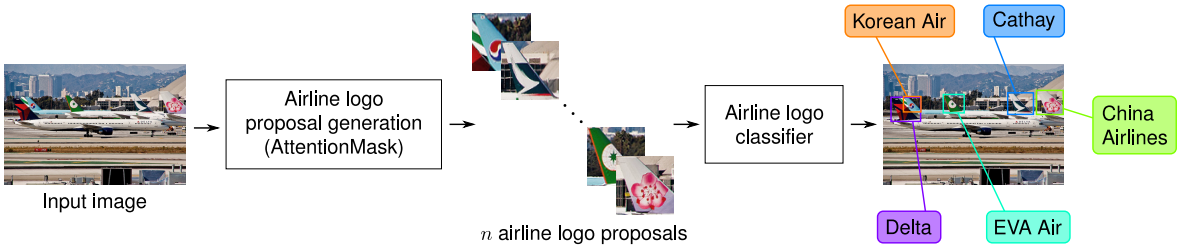


Figure 9.3: Overview of our proposed airline logo detection system based on tailored components. In the first step (left), an adapted AttentionMask system generates airline logo proposals. Subsequently, these proposals (center) are classified with our novel lightweight airline logo classifier to remove background proposals and determine the airline class (right). Note that the classes *Korean Air*, *Delta*, and *China Airlines* are not part of our PSLogos dataset and are only included for visualization. Input image taken from Lin et al. [2014].

9.1.3 Airline Logo Detection System

In this section, we introduce our tailored airline logo detection system. The system, visualized in Fig. 9.3, is separated into two distinct components for generating airline logo proposals and classifying the proposals. To generate airline logo proposals, we use a variation of AttentionMask (see Ch. 4) adapted to the airline logo detection task. Subsequently, we apply our tailored airline logo classifier to the first n airline logo proposals. The lightweight classifier assigns each proposal to an airline logo class or the background class. The subsequent sections describe the individual components in more detail. For our learning-free data augmentation strategy, see Sec. 9.1.4.

Airline Logo Proposal Generation

For generating airline logo proposals, we utilize our AttentionMask system described in Ch. 4. As shown in the evaluation in Sec. 4.4, AttentionMask generates strong object proposal generation results across all object sizes. Applying SAM (see Ch. 5) is not necessary in this case, since we need box proposals for the subsequent classification and do not rely on precise segmentation masks. To switch from general object proposal generation to airline logo proposal generation, we adapt AttentionMask in three ways. First, we remove the pyramid levels $\mathcal{S}_8$ and $\mathcal{S}_{16}$ from AttentionMask's feature pyramid, which are designed to discover small objects. Levels for small objects are not necessary, since the airline logos in our training set are medium or large. Hence, $\mathcal{S}_8$ and $\mathcal{S}_{16}$ cannot learn to generate proposals leading to false positives and reducing the computational efficiency.

Second, we change the output of AttentionMask from masks to boxes to match the demands of our CNN-based classifier by generating tight bounding boxes around the generated masks. Similarly, we change the targets for the segmentation from masks to boxes to train the system. Third, we only utilize the best 10 proposals according to AttentionMask's objectness score and apply NMS since the number of airline logos per image is small. Finally, we train the new AttentionMask variation on the PSLogos dataset with the same training regime described in Sec. 4.3. Hence, we do not use weights pre-trained on the COCO dataset. We refer the reader to Sadeghi [2019] for more details on the optimization of AttentionMask's structure and training.

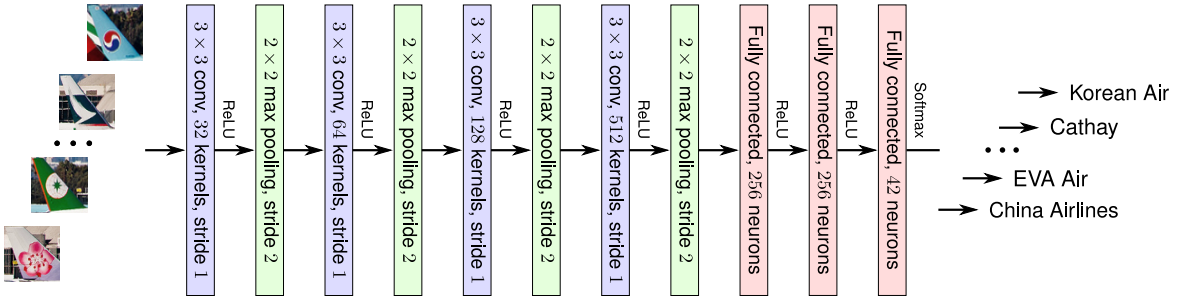


Figure 9.4: Architecture of our proposed airline logo classifier with four convolutional layers and three fully connected layers. Note that the classes *Korean Air* and *China Airlines* are not part of our PSLogos dataset and are only included for visualization. Input images taken from Lin et al. [2014].

Airline Logo Classification

To classify the airline logo proposals, we tailor an airline logo classifier to reflect the differences between objects and logos. In this section, we give an overview of the study leading to our airline logo classifier and subsequently present this classifier in more detail.

Study on the Airline Logo Classifier To tailor a classifier for airline logo detection, we conduct a structured architecture search based on the general architecture of VGG nets [Simonyan and Zisserman, 2015]. We choose VGG nets since VGG-19 leads to better airline logo classification results than other standard image classification approaches like ResNet-50 [He et al., 2016a] and InceptionV3 [Szegedy et al., 2016] as we will show in Sec. 9.1.5. VGG net architectures follow simple design principles (see Appendix A.1). Layers are grouped in stages that conclude with a max pooling layer. All layers within a stage have the same number of kernels, while the number of kernels increases from stage to stage. We follow these principles and vary the number of stages, the number of layers per stage, the number of fully connected layers, and the number of kernels or neurons per layer. Overall, we train and evaluate 11,218 small CNNs based on the annotated boxes of the PSLogos dataset with simple data augmentations (horizontal flipping and rotation). All annotated boxes were resized to 64×64 pixels for training and evaluation. Based on the study results, we select the classifier with the best classification performance. For more details on the study, see Heid [2018].

Final Airline Logo Classifier As the result of the study, we propose our tailored airline logo classifier based on the VGG-style architecture. Figure 9.4 presents the structure of the classifier. It consists of four stages with one convolutional layer per stage followed by three fully connected layers. The final fully connected layer has 42 neurons representing the 41 airline classes in the PSLogos dataset and the background class. This lightweight design leads to an effective and efficient classification and is in line with the findings of Bianco et al. [2017] on logo classification. To integrate the airline logo classifier into the overall system, we resize the airline logo proposals to size 64×64 and apply the classifier to each proposal. Hence, the classifier assigns each proposal to one of the 41 airline classes or the background class.

(a) Fog **(b)** Heavy clouds/rain **(c)** Raindrops

Figure 9.5: Examples of our proposed data augmentation strategy. The first row depicts clean image patches, while the second row visualizes the results of the data augmentations applied to the clean image patches. For comparison, the final row shows images patches with real effects of fog (a), heavy clouds or rain (b), and raindrops (c). Base images taken from Lin et al. [2014] (first two rows) and Heid [2018] (final row).

9.1.4 Data Augmentation

To increase the robustness of our airline logo detection system w.r.t. the effects of adverse weather conditions, we present a learning-free data augmentation strategy. The data augmentations imitate the effects of fog, heavy clouds or rain as well as raindrops in the PS-HAM images, which are not represented in the training data (PS-TRAIN). Since no training data with adverse weather effects is available in our dataset, our data augmentation is learning-free. This is different from most methods that handle adverse weather conditions in object detection (see Sec. 9.1.1).

The first augmentation addresses fog that leads to a brighter image with reduced contrast. To mimic this effect, we brighten each image with a random brightness component between 0.2 and 1. The color value of each pixel in the image is raised to the power of the brightness component. Note that we expect color values in the range of $[0, 1]$. The application of this augmentation leads to artificial fog reducing the contrast as visible in the second row in Fig. 9.5(a). To model the effects of heavy clouds or rain, we apply standard contrast reduction with random strength to the images. This reduces the contrast without brightening the image by combining each pixel's color values with the mean color values of the image using a random blending factor. The results of this data augmentation and the effects of real heavy clouds or rain are visible in Fig. 9.5(b) (second row vs. third row).

Finally, to imitate the effect of raindrops on a window between the camera and the airplane, we apply local Gaussian blur to the image. We employ the blur by creating a blurred version of the image using a Gaussian filter and extracting 25 to 125 circular areas of random diameter from the blurred image. These circular, blurred image patches are the artificial raindrops and replace the same area in the original image. This leads to local distortions as created by natural raindrops. The second row in Fig. 9.5(c) visualizes the effect of the artificial raindrops on clean images, while the third row in Fig. 9.5(c) depicts actual raindrops.

For training the two components of our airline logo detection system, we randomly apply one of the three augmentations to each PS-TRAIN image. Overall, the simple learning-free data

augmentation strategy imitates fog, dark clouds or rain, and raindrops increasing the system's robustness to such effects as we will discuss in the evaluation.

9.1.5 Experiments

To assess the quality of our proposed airline logo detection system, we evaluate the system on our PSLogos dataset. Additionally, this evaluation investigates the differences between using application-agnostic object detectors and a tailored system for logo detection as well as the influence of adverse weather conditions on logo detection. Both were introduced above as major drawbacks in logo detection.

For training, we utilize the PS-TRAIN training set and report the results on the test sets PS-TEST with clean images and PS-HAM with degraded images. Unless noted, we only use standard data augmentation methods (horizontal flipping and rotation). We compare our airline logo detection system with the three application-agnostic object detection systems Faster R-CNN [Ren et al., 2016] with FPN [Lin et al., 2017b], YOLOv3 [Redmon and Farhadi, 2018], and SSD [Liu et al., 2016a], which are frequently used in logo detection. All three systems were trained on the PS-TRAIN training set, similar to our proposed system. Moreover, we use a ResNet-50 [He et al., 2016a] backbone in all systems for a fair comparison. An evaluation against other logo detection systems is impossible due to the lack of publicly available code.

To evaluate the detection quality of the systems, we follow the logo detection literature [Su et al., 2017b; Fehérvári and Appalaraju, 2019] and use *mean Average Precision* (mAP_t) [Salton and McGill, 1986; Everingham et al., 2010] with different IoU thresholds t. If no IoU threshold is given, we average between 0.5 and 0.95. The mAP_t calculates the area under the class-specific precision-recall curves followed by an averaging across the different classes to assess the detection quality. See Everingham et al. [2010] for a detailed definition of mean Average Precision. Additionally, we use Average Recall (AR) as described in Sec. 2.2.3 to assess the localization quality of the systems without the classification results.

Results on PS-TEST

First, Tab. 9.2 presents the quantitative results of the four systems on the PS-TEST test set containing clean images. From the results, it is visible that all four systems perform very well on PS-TEST. Even for precise detections ($\text{mAP}_{0.75}$), all results are above 0.8, showcasing the strong overall detection performance and the precise localization ability. Similarly, the AR for the first 10 proposals (AR@10) is high with values above 0.7. Thus, the first few proposals discover most logos precisely. When comparing the different systems, our airline logo detection system outperforms the other three systems across all measures. In terms of mAP, our system outperforms Faster R-CNN by 7.4% and SSD as well as YOLOv3 by at least 1.4%. Highlighting the precise localization quality of our system, the improvement in terms of $\text{mAP}_{0.75}$ is up to 6.5%. Similarly, our system outperforms the other systems in terms of AR@10 by 1.8% to 5.8%.

The qualitative results of our system on clean images of airplanes in Fig. 9.6 support these quantitative results. On the left of Fig. 9.6, several successful detections of our system are visualized. The images include challenging scenarios with partial occlusions, small logos, multiple instances, or challenging poses. Still, typical errors exist, which are depicted on the

Table 9.2: Results on the clean images of the PS-TEST test set in terms of mean Average Precision (mAP_t) measures using different IoU values t and Average Recall for the first 10 proposals (AR@10). **Bold font** highlights the best results.

Method	mAP↑	$mAP_{0.50}$↑	$mAP_{0.75}$↑	AR@10↑
SSD	0.691	0.888	0.848	0.724
YOLOv3	0.698	0.898	0.869	0.733
Faster R-CNN	0.659	0.925	0.826	0.705
Ours	**0.708**	**0.929**	**0.880**	**0.746**

Figure 9.6: Qualitative results of our proposed airline logo detection systems on clean images of airplanes. The green boxes denote all successful detections per image with the predicted class. Red boxes denote all other detections that do not match airline logos on airplane tails. Input images taken from Lin et al. [2014].

right of Fig. 9.6. Both cases show that our airline logo detection system mistakes parts of the airplane like the wing or the winglets for an airplane tail. This is reasonable given the similar shape and, in the case of the winglet, even a similar coloring. Nevertheless, these results are unwanted and impair the detection performance.

After highlighting the strength of our overall system, we compare our tailored classifier to the integrated classifiers in Faster R-CNN, YOLOv3, and SSD. To this end, we reclassify the Faster R-CNN, YOLOv3, or SSD detections with our classifier. Table 9.3 presents the results of this experiment on PS-TEST. It is visible that across all three systems, the tailored classifier leads to improved detection results. For instance, the results of Faster R-CNN improve by 2.9% in terms of mAP when utilizing our classifier. Generally, the improvement is between 0.1% and 2.9% across the different measures and methods. Hence, the proposed tailored classifier is better suited for airline logo detection compared to the integrated classifiers in the object detection systems.

Overall, the results show a strong performance of all systems on the simple PS-TEST set. Nevertheless, our system outperforms all three application-agnostic object detection systems in localization, classification, and detection. This highlights the strength of a tailored logo detection system. A remaining source of errors for our system are objects that have a similar shape compared to airplane tails.

Table 9.3: Results of SSD, YOLOv3, and Faster R-CNN with the integrated classifier and our tailored classifier on the clean images of the PS-TEST test set in terms of mean Average Precision (mAP_t) measures using different IoU values t. **Bold font** highlights the best results per detection method.

Method	Classifier	mAP↑	$mAP_{0.50}$↑	$mAP_{0.75}$↑
SSD	Integrated	0.691	0.888	0.848
SSD	Ours	**0.693**	**0.912**	**0.863**
YOLOv3	Integrated	0.698	0.898	0.869
YOLOv3	Ours	**0.699**	**0.901**	**0.873**
Faster R-CNN	Integrated	0.659	0.925	0.826
Faster R-CNN	Ours	**0.678**	**0.944**	**0.850**

Table 9.4: Results on the degraded images of the PS-HAM test set in terms of mean Average Precision (mAP_t) measures using different IoU values t and Average Recall for the first 10 proposals (AR@10). DA denotes the use of our data augmentation strategy, while tiled denotes the use of the test-time tiling. **Bold font** highlights the best results without tiling, while *italic font* highlights the best overall results.

Method	DA	mAP↑	$mAP_{0.50}$↑	$mAP_{0.75}$↑	AR@10↑
YOLOv3	✗	0.101	0.157	0.129	0.104
YOLOv3	✓	0.123	0.195	0.147	0.128
Faster R-CNN	✗	0.118	0.225	0.102	0.134
Faster R-CNN	✓	0.128	0.239	0.117	0.143
Ours	✗	0.173	0.259	0.221	0.188
Ours	✓	**0.203**	**0.313**	**0.248**	**0.215**
Ours (tiled)	✓	*0.282*	*0.455*	*0.326*	*0.348*

Results on PS-HAM

Moving from the clean images in PS-TEST to the degraded images in PS-HAM, the results of all systems drop significantly as visible from Tab. 9.4. The average drop for the detection results is 80.1%, while the localization results decrease by 80.3% on average. For instance, the mAP drops by 85.5% for YOLOv3, highlighting the significant difference between images in PS-TEST and PS-HAM. Faster R-CNN and our system are slightly more robust, resulting in drops of 82.1% and 75.6% in terms of mAP. Overall, despite significantly impaired results, our airline logo detection system still outperforms all other methods by up to 80.8%. These improvements highlight the strong robustness of our system.

Applying our data augmentation scheme, which imitates adverse weather effects, leads to improved results across all systems (see Tab. 9.4). While Faster R-CNN improves by only 9.0% on average across the different measures, YOLOv3 and our proposed system improve by 20.8% and 16.2%. This shows the benefit of the data augmentation scheme supporting the detection under adverse weather conditions. Overall, our system with the proposed data augmentation scheme outperforms YOLOv3 and Faster R-CNN by up to 68.8% and

Table 9.5: Airline logo proposal generation results without classification on the validation set PS-VAL. AR denotes the Average Recall for the first, first 10, or first 100 proposals. **Bold font** highlights the best results.

Method	AR@1↑	AR@10↑	AR@100↑
FastMask	0.652	0.747	0.757
AttentionMask (Ch. 4)	0.642	0.823	0.851
Ours	**0.735**	**0.861**	**0.893**

112.0%. To further improve the results and address the different relative sizes[3] between the airline logos in PS-TRAIN and PS-HAM, we apply a test-time tiling scheme to our airline logo detection system. The tiling scheme applies our system on nine half-overlapping image tiles as well as the entire image and finally merges the ten sets of airline proposals. Our system with test-time tiling improves by another 44.4% compared to only applying the data augmentation.

In general, our system outperforms the application-agnostic object detection systems YOLOv3 and Faster R-CNN on the challenging PS-HAM test set. Moreover, the proposed data augmentation scheme improves the results of all methods on PS-HAM. However, the results of our airline logo detection system drop by 56.9% on average compared to PS-TEST even after test-time tiling and are currently far from satisfying.

Ablation Studies

After discussing the main results of the proposed airline logo detection system, we investigate three design choices of our system in more detail. We discuss the choice of our proposal generation system, the choice of our classifier, and the effects of each part of the proposed data augmentation strategy.

Airline Logo Proposal Generation We chose AttentionMask to generate airline logo proposals since it generated strong results in object proposal generation. To fit airline logo proposal generation, we adapted AttentionMask by removing two pyramid levels. The choice of AttentionMask and our adaptations are justified by the results in Tab. 9.5 comparing our adapted AttentionMask to original AttentionMask and FastMask [Hu et al., 2017a] on the PSLogos dataset. First, the results show the general improvement of original AttentionMask over FastMask (up to +12.4%). Additionally, the results indicate another improvement of up to 14.5% for the adapted AttentionMask over the original AttentionMask. Hence, choosing AttentionMask over FastMask as the base system and adapting AttentionMask to the new task lead to improved results.

Airline Logo Classification To further show the strength of our tailored airline logo classifier, we compare against the image classification systems VGG-19 [Simonyan and Zisserman,

[3]The relative size of the objects is more important than the absolute size since AttentionMask resizes the input image to a fixed height/width.

Table 9.6: Airline logo classification results without localization on the validation set PS-VAL in terms of the Top-1 Error measuring the relative number of misclassifications. The number of parameters is calculated based on 64×64 input patches. **Bold font** highlights the best result/smallest amount of parameters.

Method	Top-1 Error↓	Number of parameters
Bianco et al. [2017]	0.050	395,337
InceptionV3	0.066	23,851,784
ResNet-50	0.084	25,636,712
VGG-19	0.029	45,367,336
Ours	**0.012**	2,857,321

Table 9.7: Airline logo detection results on the degraded images of the PS-HAM test set in terms of mean Average Precision (mAP_t) measures using different IoU values t. Standard augmentations include noise, blur, and resizing, while our proposed augmentation strategy combines adding fog, heavy clouds/rain, and raindrops. **Bold font** highlights the best results.

Augmentations	mAP↑	$mAP_{0.50}$↑	$mAP_{0.75}$↑
No further augmentations	0.173	0.259	0.221
Standard augmentations	0.180	0.271	0.228
Only proposed raindrops	0.176	0.266	0.227
Only proposed fog	0.190	0.296	0.235
Only proposed heavy clouds/rain	0.193	0.305	0.236
All proposed augmentations	**0.203**	**0.313**	**0.248**

2015], ResNet-50 [He et al., 2016a], and InceptionV3 [Szegedy et al., 2016] as well as the dedicated logo classifier[4] proposed by Bianco et al. [2017]. We train all classifiers with the annotated logos in PS-TRAIN and evaluate on the annotated logos in PS-VAL. The results in Tab. 9.6 show that the proposed classifier outperforms VGG-19, ResNet-50, and InceptionV3 by 0.017 to 0.072 in terms of Top-1 Error. These results are intriguing since our classifier only uses a fraction of those systems' parameters. Moreover, our classifier also outperforms the dedicated logo classifier proposed by Bianco et al. [2017]. Overall, the results imply that a tailored classifier is better suited than application-agnostic classifiers for the problem of airline logo classification in terms of efficiency and effectiveness.

Data Augmentation Finally, we discuss the influence of the individual steps in our data augmentation strategy imitating adverse weather effects on the PS-HAM results. Additionally, we compare our data augmentation strategy to a standard data augmentation strategy that adds noise as well as blur, and resizes the image. As discussed previously, we apply horizontal flipping and rotation to augment the data in all experiments. The results of our airline logo detection system using the different data augmentations are presented in Tab. 9.7 and indicate that all utilized augmentations improve the performance. However, our complete data augmentation strategy outperforms the standard augmentations by up to 15.5% since

[4]Since no official code was published, we implemented the architecture described in the paper.

our augmentations imitate the degradations in the PS-HAM images. Thus, the proposed augmentations improve the results on PS-HAM individually and collectively outperforming standard augmentation techniques.

9.1.6 Discussion

In this section, we introduced a new dataset for airline logo detection, a tailored airline logo detection system based on AttentionMask, and a learning-free data augmentation strategy. Our new dataset comprises simple, clean images and a test set that includes challenging images degraded by adverse weather effects. This dataset fills a void in existing logo detection dataset literature, which mainly covers clean images. The proposed airline logo detection system is specifically tailored to the task of airline logo detection utilizing an adapted AttentionMask variation and a novel, lightweight classifier. The tailored design enables us to examine the differences between application-agnostic object detection systems and dedicated logo detection systems. Finally, we proposed a learning-free data augmentation strategy imitating adverse weather effects to compensate for the missing degraded training data.

Utilizing this setup, we investigated two drawbacks of logo detection systems discussed above. First, we showed that a tailored system yields better results (up to +7.4%) than application-agnostic object detectors commonly used in logo detection. Specifically, our tailored classifier outperforms all other tested classifiers while using only a few layers by taking advantage of the reduced intra-class variance of the airline logo classes. Similarly, our adapted AttentionMask variation improves the discovery of airline logos compared to application-agnostic object detection systems. Second, we showed a significant drop (up to -85.5%) in performance for all systems when moving from simple, clean images to challenging images degraded by adverse weather effects. Our learning-free data augmentation strategy counters this drop and improves the detection results by up to 24.2%. Overall, our tailored system outperforms the application-agnostic object detection systems showing solid robustness during localization and classification. Nevertheless, adverse weather effects remain a significant problem for all detection systems.

9.2 Medical Instrument Segmentation

We also apply AttentionMask to medical instrument segmentation in challenging intra-operative images acquired in a minimally invasive surgery setup. Medical instrument segmentation covers multiple formulations [Allan et al., 2019; Roß et al., 2021], while we only consider the pixel-precise segmentation of individual instrument instances. This segmentation of medical instruments during minimally invasive surgeries is a fundamental first step in pipelines for applications like automatic surgical skill assessment [Lin et al., 2019] or automated camera steering during interventions [Zhang and Gao, 2020].

Systems that segment medical instruments during minimally invasive surgeries have to deal with challenges like smoke, blood, insufficient illumination [Roß et al., 2021], or highly textured backgrounds [Pakhomov et al., 2019]. Furthermore, the types of minimally invasive surgeries vary, implying a high demand for generalization capabilities. Driven by the success of semantic segmentation and instance segmentation systems in computer vision [Long et al., 2015; Chen et al., 2017; He et al., 2017a], several CNN-based approaches for medical instrument segmentation were proposed. Tackling the pixel-precise segmentation of the individual

instrument instances, most systems apply common instance segmentation approaches [González et al., 2020; Ceron et al., 2021] such as Mask R-CNN [He et al., 2017a] or YOLACT++ [Bolya et al., 2020]. To foster further research in this area, the *ROBUST-MIS Challenge 2019* was organized [Roß et al., 2021]. Most of the participants in the challenge also proposed systems based on Mask R-CNN for the pixel-precise segmentation of the individual instruments [Roß et al., 2021].

Inspired by the success of methods based on instance segmentation, we apply our object proposal generation system AttentionMask to the task of medical instrument segmentation. To improve the results, we propose a novel, dedicated post-processing module for selecting the most promising instrument proposals. AttentionMask is well suited for medical instrument segmentation due to the strong results on the object proposal generation task, the solid robustness shown in the previous application on airline logo detection, and the class-agnostic problem formulation.

In the following subsections, we will first introduce the ROBUST-MIS Challenge 2019 as the context for applying AttentionMask to medical instrument segmentation. Subsequently, we present our adapted variation of AttentionMask, including our dedicated post-processing module, and evaluate the system on the challenge data of the ROBUST-MIS Challenge 2019. Finally, we will discuss the strengths and weaknesses of the proposed approach.

9.2.1 ROBUST-MIS Challenge 2019

Roß et al. organized the Robust Medical Instrument Segmentation (ROBUST-MIS) Challenge 2019 [Roß et al., 2021] to assess the current state of medical instrument segmentation and foster further research. The challenge allows a fair comparison of methods, introduces a large dataset with over 10.000 images, and includes images of varying complexity for a detailed assessment of the systems' capabilities. Overall, the challenge focuses on the robustness of methods to image degradations by blood, smoke, and illumination as well as the generalization abilities w.r.t. to different types of surgeries.

The challenge includes three tracks on sub-tasks of medical instrument segmentation. The first track requires a binary segmentation (instrument vs. background) of the entire image. Hence, instances are not distinguished. The second and third tracks demand a segmentation mask (track 2) or a bounding box (track 3) for each instrument. We only consider track 2 in this work since it formulates the problem as a mask-based object proposal generation task.

The challenge data [Maier-Hein et al., 2021] comprises 10.040 annotated images extracted from videos of 30 minimally invasive surgeries that cover three types of surgeries. The annotations are pixel-precise masks that label each visible medical instrument. 5.983 images of the overall dataset are assigned to the training set, while the rest is utilized for testing. The images in the test set are divided into three stages to assess the generalization capabilities of the systems. The first stage includes images from patients and surgery types that were also part of the training set, while the second stage comprises images from the same surgery types but patients that were not part of the training set. Finally, the third stage is most challenging, covering a surgery type that was unavailable for training.

Overall, the ROBUST-MIS Challenge 2019 provides a complex, publicly available dataset for medical instrument segmentation. Moreover, it presents a framework for a detailed assessment of medical instrument segmentation approaches.

9.2.2 AttentionMask for Medical Instrument Segmentation

Different from existing methods [González et al., 2020; Isensee and Maier-Hein, 2020; Ceron et al., 2021] based on instance segmentation or semantic segmentation, we consider the problem of medical instrument segmentation as an object proposal generation task. This formulation is reasonable since no classification is necessary in the context of the ROBUST-MIS Challenge 2019. Hence, we utilize our object proposal generation system AttentionMask to locate and segment medical instruments. AttentionMask is a good foundation for segmenting medical instruments, as it exhibits strong results in complex computer vision datasets and shows solid robustness w.r.t. image degradations (see Sec. 9.1.5).

In contrast to the previous application on airline logo detection, we do not change the architecture of AttentionMask. Removing levels from AttentionMask's feature pyramid is not beneficial for medical instrument segmentation in this context, since instruments of all sizes appear in the training and test data of the ROBUST-MIS Challenge 2019. For training AttentionMask, we customize the training regime by reducing the initial learning rate to 0.00001. Overall, we train AttentionMask for only 12 epochs on 80% of the training images provided by the ROBUST-MIS Challenge 2019. The remaining training images serve as a validation set.

Due to the large number of proposals that AttentionMask generates, we introduce a dedicated three-stage post-processing module. The first stage of the post-processing module discards all object proposals with a predicted objectness score below 0.8, as they are unlikely to cover any instrument. Since the remaining proposals will strongly overlay due to the dense extraction of windows in AttentionMask, we constitute groups of at least five object proposals that overlap. Groups with less than five overlapping proposals or individual proposals are disregarded, since they are unlikely to contain instruments. The third stage of our post-processing module generates one final object proposal per group by incorporating all pixels that are part of at least 10% of the group's proposals. Applying the three-stage post-processing module reduces the number of object proposals from 100 to 1.8 on average across the test set of the ROBUST-MIS Challenge 2019. Note that post-processing is only necessary for the evaluation in the ROBUST-MIS Challenge 2019 framework, since the pre-existing evaluation penalizes additionally proposed instruments. In contrast, standard evaluation frameworks in object proposal generation strongly focus on recall.

Overall, we adapt the training of AttentionMask and select the most promising proposals with a three-stage post-processing module. These steps result in a variation of AttentionMask for medical instrument segmentation.

9.2.3 Experiments

To evaluate the proposed variation of AttentionMask on medical instrument segmentation, we utilize the complex dataset and the evaluation framework of the ROBUST-MIS Challenge 2019. We compare our AttentionMask variation to the challenge participants [Roß et al., 2021] and the system of Ceron et al. [2021]. For our AttentionMask variation, we evaluate

three versions. The first version does not utilize the proposed post-processing module, while the second version uses the post-processing. Finally, we also evaluate a third version that represents an upper limit of our AttentionMask variation utilizing an optimal ranking of the proposals. For the optimal ranking, we only select the best matching proposal per annotated instrument.

We evaluate all methods on the test set of the ROBUST-MIS Challenge 2019. As evaluation measures, we follow the ROBUST-MIS Challenge 2019 and use the *Multi-instance Dice Similarity Coefficient* (MI_DSC) and *Multi-Instance Normalized Surface Dice* (MI_NSD). MI_DSC is based on DSC [Dice, 1945] and assesses the quality of a proposal compared to an annotated instrument based on the overlap of the areas. The MI_NSD, based on NSD [Nikolov et al., 2018], only considers the overlap of the boundary regions. To better assess the robustness of the methods, we report the 5% quantile for both scores as in Roß et al. [2021]. Hence, only the worst-case scenarios are considered to better evaluate the applicability in clinical practice. Additionally, we present quantitative results in terms of Average Recall (AR, see Sec. 2.2.3), similar to the other experiments presented throughout this thesis. For the AR-based evaluation, we only evaluate the version of our AttentionMask variation without the dedicated post-processing module to better match the typical evaluation framework in object proposal generation.

Evaluation in the ROBUST-MIS Challenge 2019 Framework

First, we compare our AttentionMask variation to the participants of the ROBUST-MIS Challenge 2019 and the system of Ceron et al. [2021]. Table 9.8 presents the results on the most challenging third stage of the ROBUST-MIS Challenge 2019 test set in terms of MI_DSC and MI_NSD. The results show that our AttentionMask variation with the dedicated post-processing module outperforms two challenge participants in terms of MI_DSC and three challenge participants in terms of MI_NSD. Additionally, the results show a strong positive influence of the post-processing on the results. However, the results are well below the best system (-54.8% in terms of MI_DSC), the participant *www* from the ROBUST-MIS Challenge 2019. Assuming an optimal ranking of our object proposals selects only the relevant proposals without duplicates, the results change drastically. Given this ranking, our AttentionMask variation outperforms *www* by 67.8% in terms of MI_DSC and 74.3% in terms of MI_NSD. These results showcase the great potential of our AttentionMask variation for medical instrument segmentation but also indicate the subpar quality of the original ranking.

Note that the results in Tab. 9.8 present 5% quantiles to assess the worst-case performance. Therefore, MI_DSC or MI_NSD score of 0.00 do not imply that those methods are unable to segment any medical instrument across the entire dataset. See Roß et al. [2021] for complete results of the challenge participants.

Evaluation in the Object Proposal Generation Framework

To further investigate the suitability of our AttentionMask variation in the context of medical instrument segmentation, we evaluate the system without the dedicated post-processing module in an object proposal generation framework. First, we assess the generalization capabilities of our proposed AttentionMask variation across the three stages of the ROBUST-MIS Challenge 2019 test set in terms of AR. The results in Tab. 9.9 show that our system

Table 9.8: Results on stage 3 of the ROBUST-MIS Challenge 2019 test set in terms of Multi-instance Dice Similarity Coefficient (MI_DSC) and Multi-Instance Normalized Surface Dice (MI_NSD). All results are 5% quantiles. The first part includes the participants of the ROBUST-MIS Challenge 2019, while the second part covers an independent system. All numbers for other systems were taken from Roß et al. [2021] and Ceron et al. [2021]. **Bold font** highlights the best results, while *italic font* indicates the best results using an optimal proposal ranking in our system.

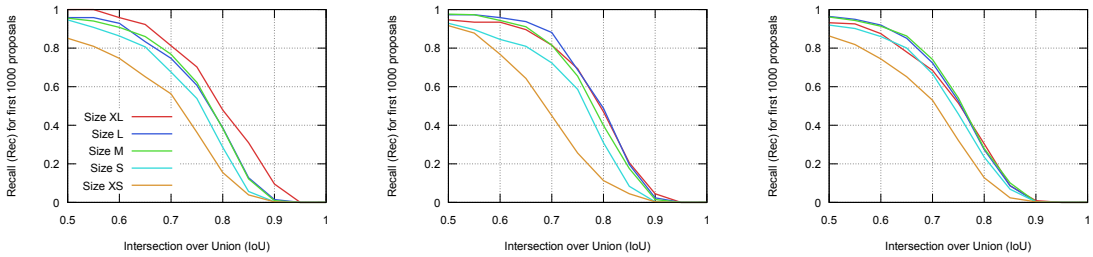
Participant/Method	MI_DSC↑	MI_NSD↑
VIE	0.00	0.00
caresyntax	0.00	0.00
fisensee [Isensee and Maier-Hein, 2020]	0.17	0.16
CASIA_SRL	0.19	0.27
SQUASH	0.22	0.26
Uniandes	0.26	0.29
www	**0.31**	**0.35**
Ceron et al. [2021]	**0.31**	0.34
Ours (without post-processing module)	0.00	0.00
Ours (with post-processing module)	0.14	0.19
Ours (with optimal ranking)	*0.52*	*0.61*

Table 9.9: Results of our proposed AttentionMask variation without the post-processing module on the different stages of the ROBUST-MIS Challenge 2019 test set in terms of three Average Recall (AR) measures.

Test set stage	AR@1↑	AR@10↑	AR@100↑
Stage 1	0.182	0.471	0.533
Stage 2	0.214	0.502	0.554
Stage 3	0.156	0.420	0.497

generalizes well to unseen patients (stage 2) and the unseen surgery type (stage 3). In terms of AR@10, the results vary between 0.420 and 0.507 for the different stages. Interestingly, the results on stage 2 are better than those on stage 1, although the patients in stage 1 have been part of the training set. This highlights our system's strong generalization capabilities and is different from the findings presented by Roß et al. [2021]. Moreover, the results in Tab. 9.9 indicate that beyond the first 10 proposals, no major improvement is evident. Hence, the rough ranking in our AttentionMask variation produces good results, while the detailed ranking of the best proposals is suboptimal as shown above.

We further investigate the results of our AttentionMask variation w.r.t. the relative sizes of instruments. This is similar to the evaluation based on size-specific measures AR^S, AR^M, and AR^L (see Sec. 2.2.3). To fit the ROBUST-MIS Challenge 2019 data, we define five relative size ranges: instruments covering less than 1% (XS), 1%-2% (S), 2%-5% (M), 5%-10% (L), and more than 10% (XL) of the image. Figure 9.7 presents the size-specific Recall (Rec) for our system across different IoU levels on the three stages of the ROBUST-MIS Challenge 2019 test set. Across all stages, the Rec is only slightly affected by the relative size. A substantial drop in results is visible only for tiny instruments (XS) at medium IoU levels. Hence, due to AttentionMask's focus on small objects, there is no major difference between discovering



(a) Stage 1 **(b)** Stage 2 **(c)** Stage 3

Figure 9.7: Recall (Rec) for the first 1000 proposals of our AttentionMask variation without the post-processing module across the three stages of the ROBUST-MIS Challenge 2019 test set. The results are split w.r.t. to different relative sizes of the annotated instruments. Size XS includes instruments covering less than 1% of the image area, size S includes instruments covering 1%-2% of the image area, size M includes instruments covering 2%-5% of the image area, size L includes instruments covering 5%-10% of the image area, and size XL includes instruments covering more than 10% of the image area.

tiny, small, medium, large, and extra large instruments. This behavior is different from the findings of Roß et al. [2021] during the ROBUST-MIS Challenge 2019.

Further investigating the robustness, Fig. 9.8 depicts qualitative results of our AttentionMask variation without the post-processing module in four challenging scenarios from the ROBUST-MIS Challenge 2019 test data. Despite challenging conditions, our AttentionMask variation segments most instruments across the images. The results in the first three rows indicate that our system properly handles multiple instruments (first row), smoke (second row), or poor illumination (third row). These observations are in line with the findings of Roß et al. [2021] during the ROBUST-MIS Challenge 2019. However, some challenging scenarios persist as the result in the final row indicates. For instance, partially occluded instruments that lead to disconnected instrument parts on the image plane are not handled well by our system. Similar observations were made by Roß et al. [2021] during the ROBUST-MIS Challenge 2019. Additionally, our system also misses the instruments' tips on some occasions (second row). These errors are related to the coarse proposals of AttentionMask as discussed in Sec. 4.5.

Overall, our AttentionMask variation shows strong robustness, generalizes well and only exhibits a slight drop in results on tiny instruments.

9.2.4 Discussion

This section presented the application of AttentionMask to medical instrument segmentation in the context of the ROBUST-MIS Challenge 2019. The challenge data includes complex images acquired during minimally invasive surgeries that demand strong generalization capabilities and solid robustness to image degradations. Our adapted AttentionMask variation with the dedicated post-processing module leads to promising results in the evaluation framework of the ROBUST-MIS Challenge 2019. However, the suboptimal ranking of the object proposals in AttentionMask, which is less relevant in object proposal generation, prevents the system from outperforming state-of-the-art methods. In terms of object proposal generation measures, our AttentionMask variation shows high-quality results across all instrument sizes and strong robustness as well as generalization abilities. This is similar to the findings of Sec. 9.1.5 in the context of adverse weather conditions. Overall, the results present a very promising

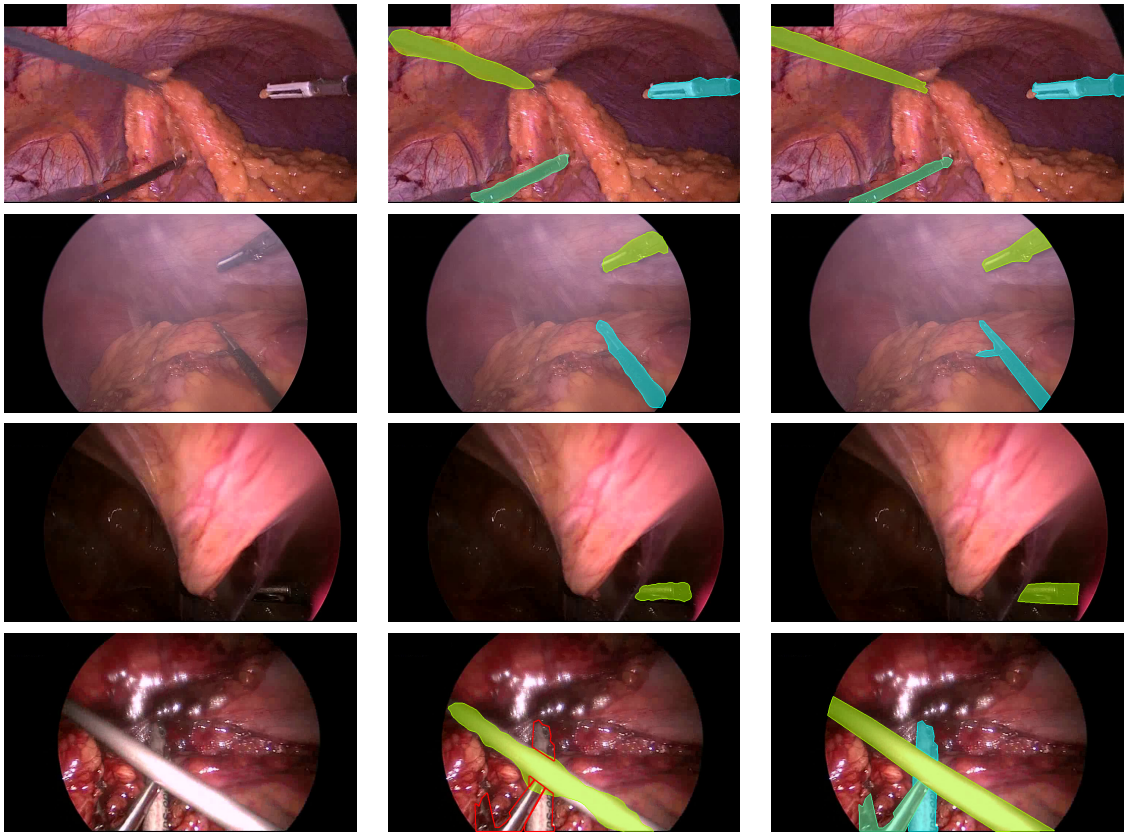


Figure 9.8: Qualitative results of our proposed AttentionMask variation without the post-processing module on stage 3 of the ROBUST-MIS Challenge 2019 test set. Filled colored contours denote located instruments, while not filled red contours denote missed instruments. Note that only the best fitting proposal (highest IoU) is visualized per annotated instrument. Input image and annotations taken from the ROBUST-MIS Challenge 2019 dataset [Roß et al., 2021; Maier-Hein et al., 2021].

foundation for further research on medical instrument segmentation that focuses on improving the ranking of object proposals.

9.3 Apple Localization in Orchard Environments

Finally, we utilize AttentionMask for apple localization in complex orchard environments. Apple localization or fruit localization/detection is regarded as a fundamental first step in (semi-)automated yield estimation or fruit picking [Sa et al., 2016; Koirala et al., 2019b; Häni et al., 2020]. For instance, yield estimation nowadays heavily relies on historical data and manual sampling, making the process labor-intensive and inaccurate [Wang et al., 2013b; Anderson et al., 2019]. By automating the sampling process, the reliability of such estimations could increase and free up human resources [Anderson et al., 2019].

The *MinneApple* dataset [Häni et al., 2020] offers a publicly available dataset for apple localization and counting with 1000 images of apple trees captured in complex orchards environments (see Fig. 9.9). Some of the major challenges in such a setting are the high level of clutter induced by the leaves, the small relative size of the apples, and the partial occlusions

Figure 9.9: Example images from the MinneApple dataset [Häni et al., 2020] depicting apples in complex orchard environments. Challenges like a large amount of clutter or the small relative size of the apples are well visible. Base images taken from the MinneApple dataset [Häni et al., 2020].

of several apples. Moreover, apples in the MinneApple dataset appear in different degrees of ripeness, are clustered together, and are illuminated differently across the trees due to the uncontrolled outdoor environment. Across the dataset, 41,325 apples are manually annotated with pixel-precise masks. Overall, the MinneApple dataset presents realistic, complex images for the task of apple localization.

To localize or detect fruits in such challenging scenarios, several systems have been proposed. Mostly, they rely on application-agnostic object detection [Sa et al., 2016; Bargoti and Underwood, 2017; Koirala et al., 2019b] or instance segmentation systems [Yu et al., 2019] like Faster R-CNN [Ren et al., 2016] or Mask R-CNN [He et al., 2017a]. Koirala et al. [2019a] present a review of recent approaches. Since the classification of the fruits is not necessary, systems that only localize objects are a natural fit for fruit localization or fruit detection. Hence, we apply our object proposal generation system AttentionMask to the problem of apple localization in the context of the complex MinneApple dataset. AttentionMask is well-suited for this application since it has shown promising results on small objects (see Sec. 4.4.1). To locate the tiny apples on the trees, we propose two variations of AttentionMask based on an extended feature pyramid and a tiling mechanism. Finally, our evaluation on the MinneApple dataset shows the benefits of our proposed variations on small and tiny apples in cluttered, complex environments.

9.3.1 Localizing Apples with AttentionMask

The examples in Fig. 9.9 show that the apples depicted in the MinneApple dataset are mainly small or tiny. Overall, more than half of the apples are smaller than 22.5^2 pixels. Hence, we apply AttentionMask to the problem of apple localization on the MinneApple dataset since it has shown strong results in discovering small objects. However, several apples are too small to fit the finest pyramid level $\mathcal{S}_8$ in AttentionMask. Therefore, we propose two variations of AttentionMask for apple localization on the MinneApple dataset. While the first variation extends the feature pyramid level, the second utilizes a tiling strategy.

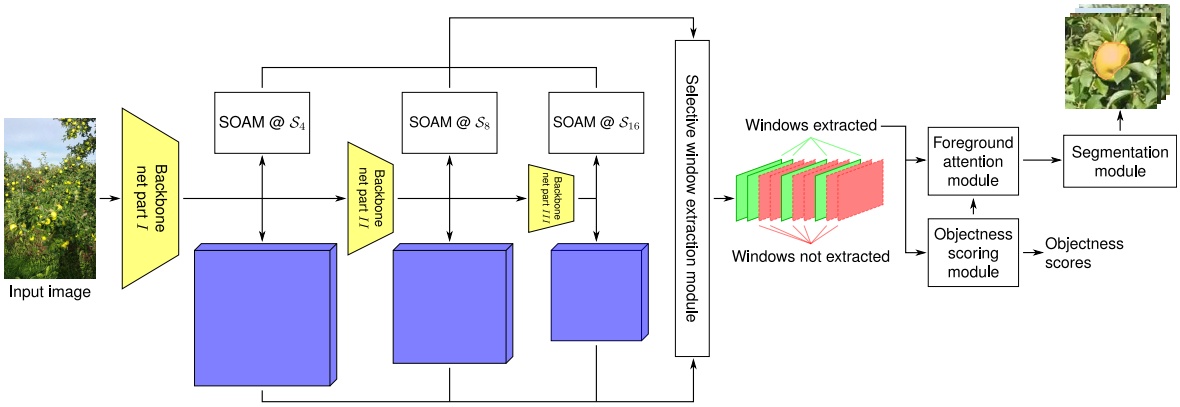


Figure 9.10: Overview of the proposed AttentionMask$_4^{16}$ system for apple localization, based on AttentionMask. Unlike original AttentionMask, we split the first part of the backbone, yielding a pyramid level $\mathcal{S}_4$ to improve the localization of tiny apples. Moreover, we remove all pyramid levels above $\mathcal{S}_{16}$. The system's remaining parts, including SOAMs, window extraction, objectness scoring, and segmentation, are identical to the original AttentionMask. Input image taken from the MinneApple dataset [Häni et al., 2020].

AttentionMask$_4^{16}$

The most natural way to enable AttentionMask to localize even smaller apples is the addition of a pyramid level $\mathcal{S}_4$ as the new feature pyramid's base level. In contrast to $\mathcal{S}_8$, the new level $\mathcal{S}_4$ enables AttentionMask to localize tiny apples since the feature map at this level is only downsampled by a factor of 4 w.r.t. the input image. To add $\mathcal{S}_4$, we split the first part of the backbone network similar to the process described in Sec. 4.2.3 to introduce $\mathcal{S}_8$. Consequently, the backbone network is split into three parts as visualized in Fig. 9.10 and *conv2* features (see Appendix A.2) are utilized for $\mathcal{S}_4$. Although these features are not as semantically rich as the features from the deeper pyramid levels, they are sufficient since the apples have a simple appearance.

On the other end of the feature pyramid, we remove all pyramid levels above $\mathcal{S}_{16}$, since no apple in the training set is large enough to match a pyramid level above $\mathcal{S}_{16}$. These modifications lead to AttentionMask$_4^{16}$ as visualized in Fig. 9.10. All remaining components and the training regime explained in Sec. 4.3 are kept.

Tiled AttentionMask

Besides introducing the new pyramid level $\mathcal{S}_4$, increasing the resolution of the input images is another way to improve the localization of tiny apples. Hence, we tile the input image resulting in 26 overlapping tiles covering the entire image similar to Bargoti and Underwood [2017], Koirala et al. [2019b], and Häni et al. [2020]. The tiles have a third of the original image's width and a quarter of the original image's height. This tiling allows a more detailed processing of the original image, since AttentionMask resizes the input image to a fixed height/width. Different from the test-time tiling for airline logo detection in Sec. 9.1.5, we also utilize the tiling during training since training data from the same size distribution as the test data is available.

The AttentionMask system itself is unchanged. Hence, we apply standard AttentionMask to each tile. Finally, we merge the results, apply NMS and retain the best 100 proposals

according to the objectness score. For training the system, we use the same training regime as described for AttentionMask but reduce the learning rate to 0.00007. We denote this system as Tiled AttentionMask.

9.3.2 Experiments

To assess the proposed AttentionMask variations, we evaluate the systems on the challenging MinneApple dataset [Häni et al., 2020]. The MinneApple dataset comprises 1000 images, however, while only 670 images include publicly available annotations. Therefore, we randomly split the 670 images into 600 images for training, 40 images for validation, and 30 images for testing. We follow the standard evaluation pipeline in object proposal generation and use Average Recall (AR, see Sec. 2.2.3) to assess how many apples are localized and how well they are localized. To better evaluate the performance on the tiny apples, we introduce a new variation of the size-specific AR. In contrast to the original definition of AR^S, which considers all apples smaller than 32^2 pixels, we redefine AR^S to consider only the size range from 22.5^2 pixels to 32^2 pixels. All apples smaller than 22.5^2 pixels are incorporated into the new AR^{XS}.

Besides our two variations of AttentionMask, we evaluate original AttentionMask and FastMask since both generate high-quality results on the object proposal generation task as demonstrated in Sec. 4.4.1. Additionally, we apply the proposed tiling to FastMask (Tiled FastMask), showing the general applicability of the idea.

Quantitative Results

Table 9.10 presents the quantitative results of AttentionMask and FastMask, $\text{AttentionMask}_4^{16}$, Tiled AttentionMask, and Tiled FastMask on our MinneApple test set. The results show that FastMask is unable to locate any apples, since it lacks a pyramid level $\mathcal{S}_8$. In contrast, AttentionMask locates several apples, leading to an AR@100 of 0.243. Adding the new pyramid level $\mathcal{S}_4$ ($\text{AttentionMask}_4^{16}$) improves the results of AttentionMask by 28.0% in terms of AR@100. Applying the introduced tiling to AttentionMask (Tiled AttentionMask) further improves the results (+ 25.1%), leading to an AR@100 of 0.415. Especially on tiny apples (AR^{XS}@100), Tiled AttentionMask exhibits substantial improvements over AttentionMask (+ 133.3%) and $\text{AttentionMask}_4^{16}$ (+ 92.1%). Applying the tiled processing to FastMask (Tiled FastMask) enables the system to locate apples as well. However, the results stay below the level of Tiled AttentionMask. Despite its strong performance, it is important to note that the tiling significantly increases the GPU runtime. For Tiled AttentionMask, the GPU runtime is more than 15 times higher compared to $\text{AttentionMask}_4^{16}$.

Overall, the results are strongly related to the size of the apples. The drop from small apples (AR^S@100) to tiny apples (AR^{XS}@100) is 57.9% on average. Hence, despite the efforts presented in Sec. 9.3.1, tiny apples are still substantially more challenging to localize compared to small ones.

Qualitative Results

To better assess the performance of the different systems, Fig. 9.11 presents qualitative results of AttentionMask, $\text{AttentionMask}_4^{16}$, Tiled FastMask, and Tiled AttentionMask. Overall,

Table 9.10: Results of AttentionMask, FastMask, and the proposed variations on our MinneApple test set in terms of five Average Recall (AR) measures. AR^{XS}, AR^{S}, and AR^{M} denote results on tiny, small, and medium apples. Note that large apples do not exist in the MinneApple dataset. The first part of the table covers standard object proposal generation methods, while the second and third part consist of the proposed variations. **Bold font** highlights the best results.

Method	AR@10↑	AR@100↑	AR^{XS}@100↑	AR^{S}@100↑	AR^{M}@100↑
AttentionMask (Ch. 4)	0.071	0.243	0.126	0.336	0.337
FastMask	0.000	0.000	0.000	0.000	0.000
$\text{AttentionMask}_4^{16}$	**0.099**	0.311	0.153	0.386	**0.570**
Tiled FastMask	0.061	0.354	0.284	0.423	0.374
Tiled AttentionMask	0.073	**0.415**	**0.294**	**0.500**	0.549

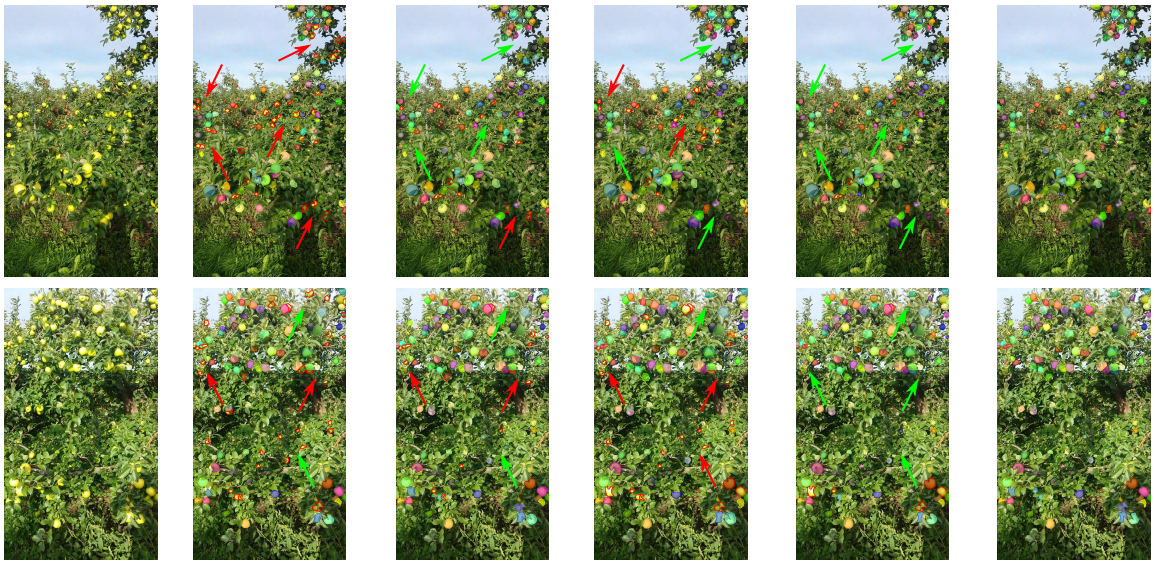


Figure 9.11: Qualitative results of AttentionMask, $\text{AttentionMask}_4^{16}$, Tiled FastMask (T-FastMask), and Tiled AttentionMask (T-AttentionMask) on our MinneApple test set. Filled colored contours denote located apples, while not filled red contours denote missed apples. Note that only the best fitting proposal (highest IoU) is visualized per annotated apple. The red arrows denote groups of missed apples, while the green arrows highlight the successful localization of those apples. Input images and annotations taken from the MinneApple dataset [Häni et al., 2020].

many apples are localized by the four systems, while even humans struggle to locate some of the apples, which can be seen by several smaller apples not annotated in the ground truth. Despite a low contrast between the yellow apples and the green leaves, the varying illumination, or the clutter introduced by the omnipresent leaves, AttentionMask$_4^{16}$ and Tiled AttentionMask locate most apples in both examples. Typical failure cases for our variations are related to leaves that severely occlude apples. Tiled FastMask is also able to locate most apples. However, it still misses more tiny apples than Tiled AttentionMask due to the missing pyramid level $\mathcal{S}_8$. In contrast, AttentionMask also misses several small or partially occluded apples, which are mostly located by our variations. Overall, the qualitative results support the quantitative results and highlight the improvements of the proposed AttentionMask variations.

9.3.3 Discussion

In this section, we applied AttentionMask to apple localization in apple orchard environments based on the MinneApple dataset. The dataset is challenging since the images depict complex scenarios with a high level of clutter (leaves) as well as tiny and partially occluded apples. To improve the discovery of tiny apples, we proposed two variations of AttentionMask based on an extended feature pyramid and a tiling strategy. The evaluation showed that our variations outperform the original AttentionMask and FastMask, especially on tiny apples. While the tiling strategy results in a better localization performance, the extended feature pyramid is computationally less demanding. Overall, both variations are able to locate a substantial amount of apples in the complex MinneApple dataset. These results again highlight the ability of AttentionMask to discover small objects and perform well in challenging application scenarios.

9.4 Discussion

This chapter presented three applications of our object proposal generation system AttentionMask to relevant real-world tasks. All three applications involve uncontrolled environments, leading to several challenges. In the airline logo detection task, mainly adverse weather conditions pose a major challenge to AttentionMask. The medical instrument segmentation data also suffers from multiple image degradations and demands strong generalization capabilities across different surgery types. Finally, the apple localization in complex orchard environments requires locating tiny objects in cluttered scenes with high similarity between target objects (apples) and clutter (leaves).

Despite these challenges, the proposed AttentionMask variations produce strong results showcasing the versatility of AttentionMask. For instance, our tailored AttentionMask-based airline logo detection system outperforms all other systems on the airline logo detection task in our evaluation. Additionally, our system shows solid robustness w.r.t. the adverse weather effects even without our proposed data augmentation. On the medical instrument segmentation task, our AttentionMask variation produces very promising results that would outperform all competitors given a better ranking of the object proposals. The promising results are highlighted by our system's strong robustness and generalization abilities. Finally, on the task of apple localization in complex orchard environments, our AttentionMask

variations show strong results on small and tiny objects despite a substantial amount of clutter.

Overall, the three applications show AttentionMask's versatility to produce strong results across various image domains and under diverse challenging conditions outside typical object proposal generation datasets. The main strengths are AttentionMask's flexibility, robustness, and a strong performance on small and tiny objects. Still, challenging scenarios, including the ranking of proposals remain.

Chapter 10

Conclusion

Table of Contents

This thesis presented novel approaches for object proposal generation that effectively address limitations of previous approaches. After summarizing the thesis's main contributions in Sec. 10.1, we discuss important strengths and limitations of our newly developed methods in Sec. 10.2. The thesis concludes in Sec. 10.3 with a discussion of potential directions for future research based on the main results of this thesis.

10.1 Summary

In the introduction of this thesis, we defined object proposal generation as the class-agnostic discovery of objects and identified two major limitations related to existing object proposal generation methods: the discovery of small objects and the coarse segmentation masks of object proposals. Throughout this thesis, we made several methodological contributions to the areas of object proposal generation and superpixel segmentation that successfully tackle those major limitations. Moreover, we also showed the versatile applicability of our contributions and addressed three challenging application areas.

Object Proposal Generation

The discovery of small objects is one of the major challenges in object proposal generation. In modern CNN-based systems, this is mainly caused by the inherent downsampling process in CNNs that removes detailed spatial information necessary to discover small objects. To improve the discovery of small objects, we proposed the novel CNN-based object proposal generation system AttentionMask. AttentionMask is the first major methodological contribution of this thesis and utilizes the concept of visual attention to enable a computationally efficient processing pipeline. This efficient processing pipeline allows us to add a new dedicated module that improves the discovery of small objects by reducing the downsampling. In our extensive evaluation on a challenging dataset, AttentionMask outperforms all previously presented object

proposal generation methods on small objects (+52.8%[1]) and objects of all sizes (+11.5%[2]). Furthermore, AttentionMask is very efficient, showing its potential for utilization in various real-world applications.

While demonstrating a strong performance in our evaluation, AttentionMask's object proposals only loosely adhere to the object boundaries. Similar effects are visible for other CNN-based object proposal generation methods. Therefore, we addressed this limitation by introducing Superpixel-based AttentionMask (SAM) as our second major methodological contribution. SAM extends AttentionMask by utilizing a novel superpixel-based refinement that combines the initial coarse AttentionMask proposals with highly precise superpixels in an end-to-end learnable framework. While the highly precise superpixels capture fine details of objects, the initial coarse CNN-based proposals support the discovery of entire objects. This innovative combination addresses a major limitation of object proposal generation methods; the trade-off between coarse proposals with high recall and precise proposals with low recall. Moreover, our approach bridges the gap between traditional superpixel-based and modern CNN-based object proposal generation approaches. The extensive evaluation shows the strong boundary adherence of the object proposals generated with SAM and FH superpixels (SAM+FH) as well as the improved overall object proposal generation results (+28.0%[3]) on challenging datasets.

As a final contribution in the area of object proposal generation, we identified four object properties that negatively impact the performance of object proposal generation methods: (1) the objects' size, (2) the objects' aspect ratio, (3) the shape complexity of the objects, and (4) the objects' contrast. These findings improve the understanding of existing challenges in object proposal generation.

Superpixel Segmentation

In addition to the direct contributions in the area of object proposal generation, we also presented two major methodological contributions to improve superpixel segmentations that support the object proposal generation with SAM. The thorough evaluation of SAM revealed that superpixel segmentations with less oversegmentation are beneficial for the superpixel-based refinement. To reduce the oversegmentation in modern superpixel segmentation methods, we proposed a new edge-adaptive framework for superpixel segmentation. The edge-adaptive framework adjusts the distribution of superpixels within an image to the different levels of detail based on edge detection results. This greatly reduces the oversegmentation and adapts the style of arbitrary superpixel segmentations to better fit SAM. We showed that the combination of SAM and the edge-adaptive superpixels leads to improved object proposal generation results when compared to the original non-adaptive superpixels.

The second contribution in the area of superpixel segmentation also aims to improve the results of SAM. To this end, we augmented the color-based FH superpixel segmentation method that limits oversegmentation by design with learned features to incorporate high-level semantics. This led to our novel CNN-based DeepFH superpixel segmentation method. The methods' success is fueled by the use of suitable features that are learned end-to-end in a

[1] $\text{AttentionMask}_8^{128}$ over FastMask in terms of AR^S@100 on the COCO test set.
[2] $\text{AttentionMask}_8^{128}$ over FastMask in terms of AR@100 on the COCO test set.
[3] SAM+FH over FastMask in terms of AR@100 on the LVIS test set.

pixel affinity framework and are seamlessly embedded into FH. This methodological similarity to FH leads to DeepFH superpixel segmentations that exhibit limited oversegmentation but utilize semantically rich, learned features. Combining SAM with DeepFH (SAM+DeepFH) produces high-quality object proposals in terms of recall and boundary adherence that outperform existing object proposal generation approaches by at least 32.3 %[4] on a challenging dataset.

Applications

AttentionMask was extensively evaluated on three complex real-world tasks to showcase its flexibility and robustness. First, we proposed a dedicated airline logo detection system utilizing AttentionMask and a tailored classifier. The airline logo detection task demands high robustness due to adverse weather effects that severely degrade the image quality in our data. Second, we applied AttentionMask to medical instrument segmentation in images acquired during minimally invasive surgeries. Those images are often degraded by blood, smoke, or poor illumination, leading to a strong demand for robustness. Finally, we introduced a system for localizing small and tiny apples in complex orchard environments using AttentionMask. Images captured in orchard environments are challenging due to a substantial amount of clutter. The results across all three tasks show our systems' strong performance as well as the strong robustness and flexibility of AttentionMask.

Overall, the combination of the thesis's contributions leads to a novel object proposal generation approach that innovatively unifies superpixels and CNNs to create high-quality object proposals for objects in complex environments.

10.2 Strengths and Limitations

Our object proposal generation systems AttentionMask and SAM lead to strong results on complex datasets like COCO and LVIS. These results are driven by the two major strengths of our systems. First, AttentionMask utilizes attention to substantially improve the discovery of small objects and objects of all sizes. Second, SAM significantly improves the boundary adherence of coarse AttentionMask object proposals by utilizing our highly precise DeepFH superpixels (SAM+DeepFH). This innovative combination bridges the gap between precise proposals with low recall based on superpixels and coarse proposals with high recall based on CNNs. Overall, the proposed systems advance the state-of-the-art in object proposal generation and outperform all existing object proposal generation methods by at least 32.2%[4] on challenging datasets. We hypothesize that the improved results in object proposal generation will also translate to improvements in subsequent applications like object detection, as the findings of Hosang et al. [2015] indicate.

Despite the strong performance compared to existing object proposal generation methods, some limitations remain. For instance, around 40% of the objects in the COCO dataset are

[4]SAM+DeepFH over FastMask in terms of AR@100 on the LVIS test set.

Figure 10.1: Result of SAM utilizing our DeepFH superpixel segmentations (SAM+DeepFH) on the image from the introductory example (see Fig. 1.1(a)). SAM+DeepFH successfully discovers 58 of 67 cookies, including the attended cookie from the introductory example (red cross). However, 9 cookies are still missed by the system (IoU < 0.5). Since the input image is four times larger than the images from the COCO dataset, we applied SAM+DeepFH on 9 partially overlapping tiles and recombined the results. Moreover, we applied a model trained on the COCO dataset that has no annotated cookies in the training dataset. Filled colored contours denote discovered cookies, while not filled red contours denote missed cookies. Note that only the best fitting proposal (highest IoU) is visualized per annotated cookie.

not discovered by the 100 most promising AttentionMask proposals. Moreover, our detailed evaluation shows that tiny, elongated, complex-shaped, or low-contrast objects present major challenges for our methods and all other object proposal generation methods. We, therefore, conclude that even with the availability of state-of-the-art solutions, the discovery of objects in complex scenes is still far from being solved.

Further limitations of the presented systems and object proposal generation systems in general are the impaired ranking of object proposals and the lack of real-time processing. As our evaluations show, the results of all object proposal generation systems improve when using more proposals. Hence, the ranking of the object proposals is still suboptimal as discussed in the introduction. This thesis did not try to improve the ranking, since object proposal generation systems generally focus on a high recall rather than a high precision[5] [Zitnick and Dollár, 2014; Hosang et al., 2015]. However, an impaired ranking leads to highly-ranked proposals covering background patches and degrade the results of subsequent applications like in our medical instrument segmentation approach. Therefore, improving the ranking of object proposals is still an open problem. Another remaining limitation is the processing time of object proposal generation systems. Although AttentionMask is the fastest system to generate mask-based object proposals without a dedicated backbone, its GPU runtime of 0.2s does not meet typical real-time requirements. Hence, methodological or technical improvements are necessary for object proposal generation systems to meet these requirements.

In summary, even though our proposed systems substantially improve the discovery of objects and produce high-quality results on challenging datasets, some objects in complex scenarios are still hard to discover. One such scenario is the introductory example described in Sec. 1 (see Fig. 1.1). Recall that the goal of the object proposal generation method in this context

[5]Precision in this context assesses how many object proposals discover objects.

is to discover all cookies with precise segmentation masks allowing subsequent methods to determine the attended cookie. The result in Fig. 10.1 shows that most cookies are properly discovered by our proposed system SAM using the precise DeepFH superpixel segmentations. Nevertheless, due to the challenging scene composition with occlusions and complex-shaped objects, 9 of the 67 cookies are missed by our system (IoU < 0.5) despite generally strong results on complex datasets. This highlights the demand for further improvements in object proposal generation.

10.3 Future Work

We conclude this thesis with a brief overview of potential research directions for extending the presented work.

Challenging Object Properties

Our extended evaluation in Ch. 8 revealed four novel challenges in object proposal generation. One of these challenges relates to elongated objects since they do not fit the square windows frequently used for roughly localizing possible objects. One way to mitigate this problem is to adaptively predict the aspect ratio per window [Ding et al., 2019; Wang et al., 2019]. In the case of AttentionMask and SAM, this could dilate the 10×10 grid points of a window along the image axes. A more flexible alternative would be to learn a warping of the 10×10 grid points by employing a dedicated *spatial transformer* module [Jaderberg et al., 2015]. However, the reprojection of the created proposal mask based on the warped window may not adhere well to the object boundaries.

Two further challenges, the discovery of tiny and complex-shaped objects, are related to the lack of spatial details in feature maps generated by CNN backbones. To circumvent this problem, different backbones like the *dilated residual network* [Yu et al., 2017], the *pyramid vision transformer* [Wang et al., 2021b], or the *feature pyramid network* [Lin et al., 2017b] are worth exploring. In the latter two cases, the backbone itself could replace the feature pyramid in AttentionMask, SAM, or FastMask, leading to a high-resolution feature pyramid. However, the high-resolution feature pyramid would strongly increase the number of possible windows for generating proposals. To compensate for this effect, a smart extraction policy in combination with our SOAMs has to select a small subset of high attention windows for efficient processing. Nevertheless, such an approach would require GPUs with more than 12 GB of memory, which are still rather expensive.

Superpixels and CNNs

The superpixel-based object proposal refinement in SAM leads to a better boundary adherence for object proposals as we demonstrated in our experiments. However, only up to 57.7% of the superpixel segmentations' capacities were utilized, implying that a better integration of superpixels and CNNs is necessary. In particular, the independent classification of individual superpixels in SAM could be replaced by a joint classification of all superpixels in and around a coarse proposal using *Graph Convolutional Networks* (GCNs) [Kipf and Welling, 2017]. GCNs are well-suited for this task since they are not restricted to a regular, grid-like topology and were successfully used in similar computer vision tasks like semantic segmentation [Lu et al.,

2019; Zhang et al., 2019b] or instance segmentation [Li and Gupta, 2018]. Extending this idea, a deeper integration of superpixels and CNNs in SAM is possible by replacing the window extraction with a superpixel-based GCN module. Given a high-attention location in the feature pyramid and a superpixel segmentation, the GCN module could generate a superpixel-based foreground-background segmentation of this area in one step. This integration would avoid the two-stage object proposal generation and refinement process in SAM.

Superpixel-based Refinement in Other Tasks

We designed our superpixel-based refinement system in SAM to improve coarse object proposals. However, similar issues arise in other dense prediction tasks like instance segmentation [He et al., 2017a; Liu et al., 2018], semantic segmentation [Lin et al., 2017a; Chen et al., 2017], or video object segmentation [Perazzi et al., 2017; Lu et al., 2020]. To improve the coarse results of CNN-based systems on such tasks, applying our superpixel-based refinement as a post-processing step is a promising direction. This is especially relevant for instance segmentation systems since other refinement strategies like CRFs or encoder-decoder architectures are not feasible given the large amount of generated objects. Furthermore, this integration allows class-specific superpixel segmentations, class-specific features, and a class-specific superpixel classifier, which are likely to simplify the problem formulation. For instance, classifying a superpixel as part of a giraffe or the background is easier than the class-agnostic case in object proposal generation.

Application to Video Data

This thesis addressed object proposal generation in static images. In applications like pedestrian detection [Geiger et al., 2012] or the previously discussed medical instrument segmentation [Roß et al., 2021], video data is available as well. This offers potential extensions of our systems to video data. Instead of applying our systems independently on each frame, temporal consistency could be enforced by linking the proposals between frames [Horbert et al., 2015; Ošep et al., 2020]. More challenging would be an online approach that tracks the proposals dynamically based on newly acquired frames without incorporating knowledge from future frames. This formulation is similar to online object tracking [Wu et al., 2013] but lacks an initial set of objects. Thus, proposals are generated and tracked through time until they disappear or the lack of temporal consistency disproves a proposal track. In this formulation, the recurrent updating and spawning of several proposal tracks are major challenges.

Future of Object Proposal Generation

Recently, few papers have proposed new object proposal generation approaches [Lu et al., 2018; Wang et al., 2019], while the interest in instance segmentation has grown [Lee and Park, 2020; Xie et al., 2020]. Due to the reliance of instance segmentation methods on the classification of objects, they are restricted to pre-selected object classes or the object classes seen in training. As a result, those methods do not generalize as well to objects of classes that were not part of the training data as class-agnostic methods [Pinheiro et al., 2015; Ošep et al., 2020; Kim et al., 2022]. This lack of generalization ability in instance segmentation has drawn attention recently with the introduction of the *open-world instance segmentation* task [Wang et al., 2021a; Kim et al., 2022]. The task explicitly assesses the generalization

ability by testing on object classes that are not annotated in the training data. Due to their class-agnostic nature, object proposal generation methods are a promising direction to approach this task. To further improve the generalization abilities of these methods, several directions are possible, including data augmentation [Saito et al., 2021], generation of pseudo annotations [Ahn et al., 2019; Laradji et al., 2019], moving the focus of CNNs from texture to shape [Geirhos et al., 2018a], or integrating model-based approaches like saliency [Martín García et al., 2015] or Gestalt principles [Werner et al., 2015] into CNNs. Overall, open-world instance segmentation is a promising new application area for object proposal generation systems.

Appendix A

CNN Backbone Networks in Computer Vision

Convolutional Neural Networks (CNNs) have gained much attention in computer vision since Krizhevsky et al. [2012] proposed *AlexNet* for image classification. The success was driven by a large amount of data in image classification datasets like *ImageNet* [Russakovsky et al., 2015] and the availability of powerful GPU hardware. After the seminal AlexNet many other authors proposed CNNs for image classification [Simonyan and Zisserman, 2015; He et al., 2016a; Szegedy et al., 2016]. Despite some differences, the general style of the architectures is similar. CNNs for image classification first utilize convolutional layers to extract a feature representation of input image followed by fully connected layers to predict a class. Besides image classification, systems for other computer vision tasks utilize CNNs as backbone networks to extract semantically rich features [Ren et al., 2016; Yang et al., 2016; Chen et al., 2017; Hu et al., 2017a]. On top of the backbone, such systems add task-dependent branches that utilize these features. Despite the differences between the tasks, the classification networks are still useful backbones. For instance, since they are usually pre-trained on large datasets like ImageNet, less task-specific data is necessary in a transfer learning framework.

All CNN-based systems in this thesis utilize modified versions of either *VGG* nets [Simonyan and Zisserman, 2015] or *ResNet*s [He et al., 2016a] as a backbone. Therefore, we give a brief introduction to VGG nets in Sec. A.1 and to ResNet in Sec. A.2. We will also introduce common naming conventions for the subtypes of the networks and the network parts for a simplified presentation.

A.1 VGG Nets

The seminal AlexNet [Krizhevsky et al., 2012] consists of only five convolutional layers with kernel sizes up to 11×11. The large kernel sizes lead to a larger receptive field for computing features. An alternative to large kernel sizes is a stack of multiple layers with smaller kernels sizes. Simonyan and Zisserman [2015] utilize this idea as a key component for their *Visual Geometry Group* (VGG) nets. VGG nets use only 3×3 kernels that are stacked to enlarge the receptive field. For instance, five consecutive 3×3 kernels have the same receptive field as one 11×11 while reducing the number of parameters from 122 to 50. This change leads to deeper networks with a simple architecture that only utilizes convolutional layers with 3×3 kernels and contributes to improved results [Simonyan and Zisserman, 2015].

Simonyan and Zisserman [2015] propose six networks for image classification based on this idea. All six networks share the same general structure and mainly differ in the parameters of

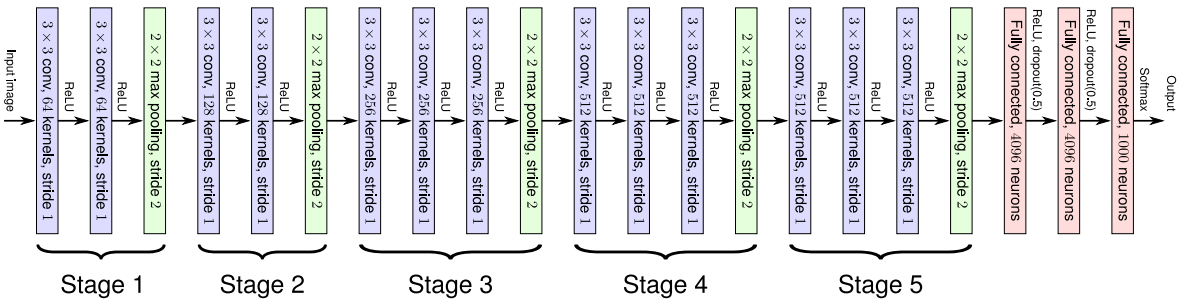


Figure A.1: Architecture of VGG-16 for image classification on the ImageNet dataset (1000 classes). Each stage consists of 2-3 convolutional layers (blue) followed by a max pooling layer (green) at the end of each stage. Finally, the learned features are classified using three fully connected layers (red).

the convolutional layers. One of these networks, VGG-16[1], serves as the backbone in many computer vision systems [Girshick, 2015; Liu et al., 2016a; He et al., 2018]. The feature extraction in VGG-16 consists of 13 convolutional layers with 3×3 kernels grouped into stages as visualized in Fig. A.1. Within each stage, the layers have a fixed number of kernels from 64 in the first stage to 512 in the final stage. Between stages, max pooling operations are introduced to downsample the feature maps. Since the VGG nets are designed for image classification, they conclude with fully connected layers and softmax. The remaining VGG nets consist of 8 to 16 convolutional layers that follow the same general structure.

Overall, VGG nets present a simple yet effective architecture for CNN backbones.

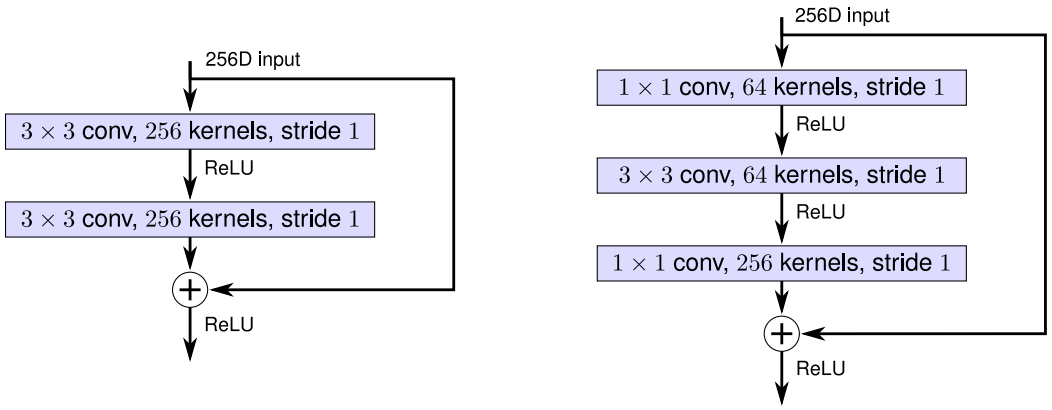
A.2 ResNet

Despite the success of VGG nets based on stacking layers, this concept leads to problems like vanishing gradients or early saturation if many layers are stacked. Vanishing gradients occur since the derivate of the loss function becomes smaller with every application of the chain rule during backpropagation in deep networks. Hence, learning becomes more difficult with every additional layer, as almost no gradient information reaches the early layers of the network[2]. Early saturation describes the effect that very deep networks with many stacked layers saturate at higher training errors than their shallower complements [He and Sun, 2015; Srivastava et al., 2015]. Thus, the extra layers impair the results, which is surprising since simple identity mappings in the extra layers would lead to results similar to the shallower networks.

He et al. [2016a] learn residual mappings that improve learning in very deep networks to address the aforementioned problems. Instead of only stacking layers, He et al. [2016a] define blocks of stacked convolutional layers with skip connections as building blocks (see Fig. A.2). The skip connection receives the same input as the first layer of the main branch and applies an identity mapping. Since the result of the skip connection is added to the result of the main branch's final layer, the main branch learns a residual mapping. This reformulation of the architecture improves learning in deep CNNs [He et al., 2016a] and serves as the fundamental

[1] Simonyan and Zisserman [2015] denote this network as architecture D.

[2] Note that other approaches to circumvent this problem exist [Glorot and Bengio, 2010; Ioffe and Szegedy, 2015].



(a) Residual block in ResNet-18 and ResNet-34 **(b)** Residual block in ResNet-50

Figure A.2: Comparison of residual blocks in ResNet-18 and ResNet-34 (a) on the one hand and ResNet-50 (b) on the other hand. While the residual block in (a) has only two layers in the main branch, the residual block's main branch in (b) has three layers (left). Both residual blocks have example hyperparameters for input and output feature maps with 256 channels (256D). Figure adapted from He et al. [2016a].

building block of the ResNet versions proposed by He et al. [2016a]. Similar to VGG nets, ResNets are used as backbone networks in many computer vision systems [Pinheiro et al., 2016; He et al., 2017a].

He et al. [2016a] propose ResNets with different numbers of learned layers n, denoted as ResNet-n. We focus our discussion on the ResNets with 18, 34, and 50 layers since they are utilized in different parts of this thesis. Other ResNets are described in He et al. [2016a] and He et al. [2016b]. The basic components of ResNet-18, ResNet-34, and ResNet-50 are the blocks with two to three convolutional layers in the main branch and a skip connection as depicted in Fig. A.2. These building blocks, residual blocks, differ between ResNet-18 and ResNet-34 on the one hand and ResNet-50 on the other hand. In ResNet-18 and ResNet-34, the residual blocks have two convolutional layers with a fixed number of 3×3 kernels (see Fig. A.2(a)). ResNet-50 consists of residual blocks with a bottleneck in the main branch, visualized in Fig. A.2(b). The bottleneck consists of a 1×1 convolutional layer to reduce the number of channels of the feature map, followed by a 3×3 convolutional layer on the reduced feature map, and a final 1×1 convolutional layer to restore the original number of channels. This structure improves the computational efficiency since the expensive 3×3 convolutional layer is applied to fewer channels. Still, the resulting feature maps have a larger capacity with more channels.

Table A.1 presents a summary of the ResNet-18, ResNet-34 and ResNet-50 architectures. In all three ResNet architectures, the residual blocks are grouped in four stages (*conv2*, *conv3*, *conv4*, and *conv5*). The remaining stage, *conv1*, only consists of a 7×7 convolutional layer. Within the other four stages, the number of kernels per residual block is fixed. Additionally, at the beginning of a stage, the feature map is downsampled by a factor of 2. Hence, the feature map at the end of the stage *conv5* represents the input image downsampled by a factor of 2^5. Since the ResNets are designed for image classification, global average pooling, a fully connected layer, and softmax conclude the networks.

Overall, ResNets allow deeper CNN backbones than VGG nets by stacking residual blocks.

Table A.1: Schematic representation of ResNet-18, ResNet-34, and ResNet-50. Convolutional layers with k kernels of size $n \times n$ are denoted as $n \times n, k$. The kernel size and stride are given for max pooling layers, while fully connected layers are specified with their number of neurons. ReLU is omitted in the representation for brevity. The blocks for ResNet-18 and ResNet-34 in the stages *conv2-conv5* are two-layer residual blocks (see Fig. A.2(a)), while the blocks for ResNet-50 denote residual blocks with bottlenecks (see Fig. A.2(b)). Table adapted from He et al. [2016a].

Stage	Downsampling factor	ResNet-18	ResNet-34	ResNet-50
conv1	×2	$7 \times 7, 64$		
		Max pooling layer (3×3, stride 2)		
conv2	×4	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3	×8	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4	×16	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
conv5	×32	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 1024 \end{bmatrix} \times 3$
		Global average pooling		
		Fully connected layer (1000)		
		Softmax		

Lists of Abbreviations, Names, and Symbols

Abbreviations and Names

AR	Average Recall (see Sec. 2.2.3)
AttentionMask	Object proposal generation method proposed in Ch. 4
BR	Boundary Recall (see Sec. 2.1.3)
BSD	Berkeley Segmentation Dataset (see Sec. 2.1.2)
CNN	Convolutional Neural Network
COB	Object proposal generation and contour detection method [Maninis et al., 2016, 2017]
COCO	Microsoft Common Objects in Context dataset (see Sec. 2.2.2)
CRF	Conditional Random Field
DeepFH	Superpixel segmentation method proposed in Ch. 7
DeepMask	Object proposal generation method [Pinheiro et al., 2015]
EA-ETPS	Superpixel segmentation method proposed in Ch. 6
EA-SLIC	Superpixel segmentation method proposed in Ch. 6
ETPS	Superpixel segmentation method [Yao et al., 2015]
Fash	Fashionista Dataset (see Sec. 2.1.2)
FastMask	Object proposal generation method [Hu et al., 2017a]
FH	Superpixel segmentation method [Felzenszwalb and Huttenlocher, 2004]
GPU	Graphics Processing Unit
GT	Ground Truth
IoU	Intersection over Union (see Sec. 2.2.3)
LVIS	Large Vocabulary Instance Segmentation dataset (see Sec. 2.2.2)
MCG	Object proposal generation method [Arbeláez et al., 2014; Pont-Tuset et al., 2017]
NMS	Non-Maximum Suppression
NYU	NYU Depth Dataset V2 (see Sec. 2.1.2)
OE	Oversegmentation Error (see Sec. 2.1.3)
OSQ	Overall Segmentation Quality (see Sec. 2.1.3)
Rec	Recall in object proposal generation (see Sec. 2.2.3)
ResNet	Residual Network (see Appendix A.2)
SAM	Superpixel-based AttentionMask, object proposal generation method proposed in Ch. 5

SAM+FH	SAM with FH superpixel segmentations (see Sec. 5.4)
SAM+DeepFH	SAM with DeepFH superpixel segmentations (see Sec. 7.3)
SAM+EA-ETPS	SAM with EA-ETPS superpixel segmentations (see Sec. 6.4)
SBD	Stanford Background Dataset (see Sec. 2.1.2)
SharpMask	Object proposal generation method [Pinheiro et al., 2016]
SLIC	Superpixel segmentation method [Achanta et al., 2012]
SOAM	Scale-specific Objectness Attention Module (see Sec. 4.2.2)
SUN	SUN RGB-D Dataset (see Sec. 2.1.2)
UE	Undersegmentation Error (see Sec. 2.1.3)

Symbols

r_{ψ_p,ψ_q}	Pearson correlation coefficient between object properties (see Sec. 8.1.3)
Ω	Set of pixels in the image
$\mathbf{p}$	Pixel as vector of x- and y-coordinates
$\mathcal{S}_n$	Level of the feature pyramid (see Sec. 4.2.1 and Sec. 5.2.1)
S	Superpixel segmentation (see Sec. 2.1.1)
S_i	Superpixel in S (see Sec. 2.1.1)

Bibliography

[Achanta et al. 2012] Achanta, R. ; Shaji, A. ; Smith, K. ; Lucchi, A. ; Fua, P. ; Süsstrunk, S.: SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 34(11), 2012

[Achanta and Süsstrunk 2017] Achanta, R. ; Süsstrunk, S.: Superpixels and polygons using simple non-iterative clustering. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Ahn et al. 2019] Ahn, J. ; Cho, S. ; Kwak, S.: Weakly supervised learning of instance segmentation with inter-pixel relations. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Ahn and Kwak 2018] Ahn, J. ; Kwak, S.: Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2018

[Alexe et al. 2010] Alexe, B. ; Deselaers, T. ; Ferrari, V.: What is an object? In: *Computer Vision and Pattern Recognition (CVPR)*, 2010

[Alexe et al. 2012] Alexe, B. ; Deselaers, T. ; Ferrari, V.: Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 34(11), 2012

[Allan et al. 2019] Allan, M. ; Shvets, A. ; Kurmann, T. ; Zhang, Z. ; Duggal, R. ; Su, Y.H. ; Rieke, N. ; Laina, I. ; Kalavakonda, N. ; Bodenstedt, S. ; Herrera, L. ; Li, W. ; Iglovikov, V. ; Luo, H. ; Yang, J. ; Stoyanov, D. ; Maier-Hein, L. ; Speidel, S. ; Azizian, M.: 2017 Robotic instrument segmentation challenge. *arXiv preprint arXiv:1902.06426*, 2019

[Altmeyer et al. 2020] Altmeyer, K. ; Kapp, S. ; Thees, M. ; Malone, S. ; Kuhn, J. ; Brünken, R.: The use of augmented reality to foster conceptual knowledge acquisition in STEM laboratory courses — Theoretical background and empirical results. *British Journal of Educational Technology (BJET)* 51(3), 2020

[Anderson et al. 2019] Anderson, N.T. ; Underwood, J.P. ; Rahman, M.M. ; Robson, A.J. ; Walsh, K.B.: Estimation of fruit load in mango orchards: Tree sampling considerations and use of machine vision and satellite imagery. *Precision Agriculture (PA)* 20(4), 2019

[Anderson et al. 2018] Anderson, P. ; He, X. ; Buehler, C. ; Teney, D. ; Johnson, M. ; Gould, S. ; Zhang, L.: Bottom-up and top-down attention for image captioning and visual question answering. In: *Computer Vision and Pattern Recognition (CVPR)*, 2018

[Arbeláez et al. 2010] Arbeláez, P. ; Maire, M. ; Fowlkes, C. ; Malik, J.: Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 33(5), 2010

[Arbeláez et al. 2014] Arbeláez, P. ; Pont-Tuset, J. ; Barron, J.T. ; Marques, F. ; Malik, J.: Multiscale combinatorial grouping. In: *Computer Vision and Pattern Recognition (CVPR)*, 2014

[Avidan and Shamir 2007] Avidan, S. ; Shamir, A.: Seam carving for content-aware image resizing. *ACM Transactions on Graphics (TOG)* 26(3), 2007

[Ban et al. 2018] Ban, Z. ; Liu, J. ; Cao, L.: Superpixel segmentation using Gaussian mixture model. *IEEE Transactions on Image Processing (TIP)* 27(8), 2018

[Bargoti and Underwood 2017] Bargoti, S. ; Underwood, J.: Deep fruit detection in orchards. In: *International Conference on Robotics and Automation (ICRA)*, 2017

[Barz et al. 2021] Barz, M. ; Kapp, S. ; Kuhn, J. ; Sonntag, D.: Automatic recognition and augmentation of attended objects in real-time using eye tracking and a head-mounted display. In: *Symposium on Eye Tracking Research and Applications (ETRA)*, 2021

[Barz and Sonntag 2016] Barz, M. ; Sonntag, D.: Gaze-guided object classification using deep neural networks for attention-based computing. In: *International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2016

[Benesova and Kottman 2014] Benesova, W. ; Kottman, M.: Fast superpixel segmentation using morphological processing. In: *Conference on Machine Vision and Machine Learning (MVML)*, 2014

[Benson and Greenberg 1969] Benson, D.F. ; Greenberg, J.P.: Visual form agnosia: A specific defect in visual discrimination. *Archives of Neurology* 20(1), 1969

[Bianco et al. 2017] Bianco, S. ; Buzzelli, M. ; Mazzini, D. ; Schettini, R.: Deep learning for logo recognition. *Neurocomputing* 245, 2017

[Bilen and Vedaldi 2016] Bilen, H. ; Vedaldi, A.: Weakly supervised deep detection networks. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Bolya et al. 2020] Bolya, D. ; Zhou, C. ; Xiao, F. ; Lee, Y.J.: YOLACT++: Better real-time instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 44(2), 2020

[Buyssens et al. 2014] Buyssens, P. ; Gardin, I. ; Ruan, S. ; Elmoataz, A.: Eikonal-based region growing for efficient clustering. *Image and Vision Computing (IMAVIS)* 32(12), 2014

[Canny 1986] Canny, J.: A computational approach to edge detection. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 8(6), 1986

[Carreira and Sminchisescu 2010] Carreira, J. ; Sminchisescu, C.: Constrained parametric min-cuts for automatic object segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2010

[Carreira and Sminchisescu 2011] Carreira, J. ; Sminchisescu, C.: CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 34(7), 2011

[Cavanagh 2011] Cavanagh, P.: Visual cognition. *Vision Research* 51(13), 2011

[Ceron et al. 2021] Ceron, J.C.A. ; Chang, L. ; Ochoa-Ruiz, G. ; Ali, S.: Assessing YOLACT++ for real time and robust instance segmentation of medical instruments in endoscopic procedures. In: *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2021

[Chaki and Dey 2020] Chaki, J. ; Dey, N.: *A beginner's guide to image shape feature extraction techniques*. CRC Pess, 2020

[Chan et al. 2019] Chan, L. ; Hosseini, M.S. ; Rowsell, C. ; Plataniotis, K.N. ; Damaskinos, S.: HistoSegNet: Semantic segmentation of histological tissue type in whole slide images. In: *International Conference on Computer Vision (ICCV)*, 2019

[Chang et al. 2011] Chang, K.Y. ; Liu, T.L. ; Chen, H.T. ; Lai, S.H.: Fusing generic objectness and visual saliency for salient object detection. In: *International Conference on Computer Vision (ICCV)*, 2011

[Chen et al. 2015a] Chen, L.C. ; Papandreou, G. ; Kokkinos, I. ; Murphy, K. ; Yuille, A.: Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: *International Conference on Learning Representations (ICLR)*, 2015

[Chen et al. 2017] Chen, L.C. ; Papandreou, G. ; Kokkinos, I. ; Murphy, K. ; Yuille, A.L.: DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 40(4), 2017

[Chen et al. 2016a] Chen, L.C. ; Yang, Y. ; Wang, J. ; Xu, W. ; Yuille, A.L.: Attention to scale: Scale-aware semantic image segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Chen et al. 2016b] Chen, T. ; Lin, L. ; Liu, L. ; Luo, X. ; Li, X.: DISC: Deep image saliency computing via progressive representation learning. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* 27(6), 2016

[Chen et al. 2015b] Chen, X. ; Ma, H. ; Wang, X. ; Zhao, Z.: Improving object proposals with multi-thresholding straddling expansion. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Chen et al. 2018] Chen, Y. ; Li, W. ; Sakaridis, C. ; Dai, D. ; Van Gool, L.: Domain adaptive Faster R-CNN for object detection in the wild. In: *Computer Vision and Pattern Recognition (CVPR)*, 2018

[Cheng et al. 2014] Cheng, M.M. ; Zhang, Z. ; Lin, W.Y. ; Torr, P.H.: BING: Binarized normed gradients for objectness estimation at 300fps. In: *Computer Vision and Pattern Recognition (CVPR)*, 2014

[Christodoulou et al. 2003] Christodoulou, C.I. ; Pattichis, C.S. ; Pantziaris, M. ; Nicolaides, A.: Texture-based classification of atherosclerotic carotid plaques. *IEEE Transactions on Medical Imaging (T-MI)* 22(7), 2003

[Chu et al. 2018] Chu, F.J. ; Xu, R. ; Vela, P.A.: Real-world multiobject, multigrasp detection. *IEEE Robotics and Automation Letters (RA-L)* 3(4), 2018

[Comaniciu and Meer 2002] Comaniciu, D. ; Meer, P.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 24(5), 2002

[Cordts et al. 2016] Cordts, M. ; Omran, M. ; Ramos, S. ; Rehfeld, T. ; Enzweiler, M. ; Benenson, R. ; Franke, U. ; Roth, S. ; Schiele, B.: The Cityscapes dataset for semantic urban scene understanding. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Criminisi 2004] Criminisi, A.: *Microsoft Research Cambridge object recognition image database.* Database, 2004

[Dai et al. 2016] Dai, J. ; He, K. ; Li, Y. ; Ren, S. ; Sun, J.: Instance-sensitive fully convolutional networks. In: *European Conference on Computer Vision (ECCV)*, 2016

[Dalal and Triggs 2005] Dalal, N. ; Triggs, B.: Histograms of oriented gradients for human detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2005

[Dice 1945] Dice, L.R.: Measures of the amount of ecologic association between species. *Ecology* 26(3), 1945

[Ding et al. 2019] Ding, J. ; Xue, N. ; Long, Y. ; Xia, G.S. ; Lu, Q.: Learning RoI transformer for oriented object detection in aerial images. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Dollár and Zitnick 2013] Dollár, P. ; Zitnick, C.L.: Structured forests for fast edge detection. In: *International Conference on Computer Vision (ICCV)*, 2013

[Dollár and Zitnick 2014] Dollár, P. ; Zitnick, C.L.: Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 37(8), 2014

[Eggert et al. 2017] Eggert, C. ; Zecha, D. ; Brehm, S. ; Lienhart, R.: Improving small object proposals for company logo detection. In: *International Conference on Multimedia Retrieval (ICMR)*, 2017

[Endres and Hoiem 2010] Endres, I. ; Hoiem, D.: Category independent object proposals. In: *European Conference on Computer Vision (ECCV)*, 2010

[Endres and Hoiem 2013] Endres, I. ; Hoiem, D.: Category-independent object proposals with diverse ranking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 36(2), 2013

[Erhan et al. 2014] Erhan, D. ; Szegedy, C. ; Toshev, A. ; Anguelov, D.: Scalable object detection using deep neural networks. In: *Computer Vision and Pattern Recognition (CVPR)*, 2014

[Everingham et al. 2010] Everingham, M. ; Van Gool, L. ; Williams, C.K.I. ; Winn, J. ; Zisserman, A.: The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision (IJCV)* 88(2), 2010

[Fan et al. 2020] Fan, D.P. ; Ji, G.P. ; Sun, G. ; Cheng, M.M. ; Shen, J. ; Shao, L.: Camouflaged object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2020

[Fan and Ling 2019] Fan, H. ; Ling, H.: Siamese cascaded region proposal networks for real-time visual tracking. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Fehérvári and Appalaraju 2019] Fehérvári, I. ; Appalaraju, S.: Scalable logo recognition using proxies. In: *Winter Conference on Applications of Computer Vision (WACV)*, 2019

[Felzenszwalb et al. 2009] Felzenszwalb, P.F. ; Girshick, R. ; McAllester, D. ; Ramanan, D.: Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 32(9), 2009

[Felzenszwalb and Huttenlocher 2004] Felzenszwalb, P.F. ; Huttenlocher, D.P.: Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)* 59(2), 2004

[Forsyth et al. 1996] Forsyth, D.A. ; Malik, J. ; Fleck, M.M. ; Greenspan, H. ; Leung, T. ; Belongie, S. ; Carson, C. ; Bregler, C.: Finding pictures of objects in large collections of images. In: *Object Representation in Computer Vision (ORCV)*, 1996

[Frintrop 2014] Frintrop, S.: *Cognitive approaches for mobile systems*, University of Bonn, Germany, Habilitation thesis, 2014

[Frintrop et al. 2014] Frintrop, S. ; Martín García, G. ; Cremers, A.B.: A cognitive approach for object discovery. In: *International Conference on Pattern Recognition (ICPR)*, 2014

[Frintrop et al. 2015] Frintrop, S. ; Werner, T. ; Martín García, G.: Traditional saliency reloaded: A good old model in new shape. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Fukunaga and Hostetler 1975] Fukunaga, K. ; Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory* 21(1), 1975

[Gadde et al. 2016] Gadde, R. ; Jampani, V. ; Kiefel, M. ; Kappler, D. ; Gehler, P.V.: Superpixel convolutional networks using bilateral inceptions. In: *European Conference on Computer Vision (ECCV)*, 2016

[Gao et al. 2017] Gao, G. ; Lauri, M. ; Zhang, J. ; Frintrop, S.: Saliency-guided adaptive seeding for supervoxel segmentation. In: *International Conference on Intelligent Robots and Systems (IROS)*, 2017

[Gao et al. 2014] Gao, Y. ; Wang, F. ; Luan, H. ; Chua, T.S.: Brand data gathering from live social media streams. In: *International Conference on Multimedia Retrieval (ICMR)*, 2014

[Geiger et al. 2012] Geiger, A. ; Lenz, P. ; Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *Computer Vision and Pattern Recognition (CVPR)*, 2012

[Geirhos et al. 2018a] Geirhos, R. ; Rubisch, P. ; Michaelis, C. ; Bethge, M. ; Wichmann, F.A. ; Brendel, W.: ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In: *International Conference on Learning Representations (ICLR)*, 2018

[Geirhos et al. 2018b] Geirhos, R. ; Temme, C.R. ; Rauber, J. ; Schütt, H.H. ; Bethge, M. ; Wichmann, F.A.: Generalisation in humans and deep neural networks. In: *Advances in Neural Information Processing Systems (NeurIPS)*, 2018

[Gerlach 2021] Gerlach, A.M.: *Applying AttentionMask to the segmentation of medical instruments*, University of Hamburg, Germany, Bachelor thesis, 2021

[Ghodrati et al. 2015] Ghodrati, A. ; Diba, A. ; Pedersoli, M. ; Tuytelaars, T. ; Van Gool, L.: DeepProposal: Hunting objects by cascading deep convolutional layers. In: *International Conference on Computer Vision (ICCV)*, 2015

[Ghodrati et al. 2017] Ghodrati, A. ; Diba, A. ; Pedersoli, M. ; Tuytelaars, T. ; Van Gool, L.: DeepProposals: Hunting objects and actions by cascading deep convolutional layers. *International Journal of Computer Vision (IJCV)* 124(2), 2017

[Gidaris and Komodakis 2016] Gidaris, S. ; Komodakis, N.: Attend refine repeat: Active box proposal generation via in-out localization. In: *British Machine Vision Conference (BMVC)*, 2016

[Girshick 2015] Girshick, R.: Fast R-CNN. In: *International Conference on Computer Vision (ICCV)*, 2015

[Glorot and Bengio 2010] Glorot, X. ; Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010

[González et al. 2020] González, C. ; Bravo-Sánchez, L. ; Arbelaez, P.: ISINet: An instance-based approach for surgical instrument segmentation. In: *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2020

[Gonzalez and Woods 2018] Gonzalez, R.C. ; Woods, R.E.: *Digital image processing*. Fourth edition. Pearson, 2018

[Goodale et al. 1991] Goodale, M.A. ; Milner, A.D. ; Jakobson, L.S. ; Carey, D.P.: A neurological dissociation between perceiving objects and grasping them. *Nature* 349(6305), 1991

[Gould et al. 2009] Gould, S. ; Fulton, R. ; Koller, D.: Decomposing a scene into geometric and semantically consistent regions. In: *International Conference on Computer Vision (ICCV)*, 2009

[Gould et al. 2008] Gould, S. ; Rodgers, J. ; Cohen, D. ; Elidan, G. ; Koller, D.: Multi-class segmentation with relative location prior. *International Journal of Computer Vision (IJCV)* 80(3), 2008

[Gu et al. 2009] Gu, C. ; Lim, J.J. ; Arbeláez, P. ; Malik, J.: Recognition using regions. In: *Computer Vision and Pattern Recognition (CVPR)*, 2009

[Gupta et al. 2019] Gupta, A. ; Dollár, P. ; Girshick, R.: LVIS: A dataset for large vocabulary instance segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Halder et al. 2019] Halder, S.S. ; Lalonde, J.F. ; de Charette, R.: Physics-based rendering for improving robustness to rain. In: *International Conference on Computer Vision (ICCV)*, 2019

[Häni et al. 2020] Häni, N. ; Roy, P. ; Isler, V.: MinneApple: A benchmark dataset for apple detection and segmentation. *IEEE Robotics and Automation Letters (R-AL)* 5(2), 2020

[Haralick et al. 1973] Haralick, R.M. ; Shanmugam, K. ; Dinstein, I.: Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics* 3(6), 1973

[Haralick and Shapiro 1985] Haralick, R.M. ; Shapiro, L.G.: Image segmentation techniques. *Computer Vision, Graphics, and Image Processing (CVGIP)* 29(1), 1985

[Harel et al. 2006] Harel, J. ; Koch, C. ; Perona, P.: Graph-based visual saliency. In: *Advances in Neural Information Processing Systems (NeurIPS)*, 2006

[Hariharan et al. 2014] Hariharan, B. ; Arbeláez, P. ; Girshick, R. ; Malik, J.: Simultaneous detection and segmentation. In: *European Conference on Computer Vision (ECCV)*, 2014

[He et al. 2017a] He, K. ; Gkioxari, G. ; Dollár, P. ; Girshick, R.: Mask R-CNN. In: *International Conference on Computer Vision (ICCV)*, 2017

[He and Sun 2015] He, K. ; Sun, J.: Convolutional neural networks at constrained time cost. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[He et al. 2016a] He, K. ; Zhang, X. ; Ren, S. ; Sun, J.: Deep residual learning for image recognition. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[He et al. 2016b] He, K. ; Zhang, X. ; Ren, S. ; Sun, J.: Identity mappings in deep residual networks. In: *European Conference on Computer Vision (ECCV)*, 2016

[He et al. 2015] He, S. ; Lau, R.W. ; Liu, W. ; Huang, Z. ; Yang, Q.: SuperCNN: A superpixelwise convolutional neural network for salient object detection. *International Journal of Computer Vision (IJCV)* 115(3), 2015

[He et al. 2018] He, X. ; Peng, Y. ; Zhao, J.: Fast fine-grained image classification via weakly supervised discriminative localization. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* 29(5), 2018

[He et al. 2017b] He, Y. ; Chiu, W.C. ; Keuper, M. ; Fritz, M.: STD2P: RGBD semantic segmentation using spatio-temporal data-driven pooling. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Heid 2018] Heid, R.: *Klassifikation von Airline-Logos auf Flugzeugleitwerken unter Verwendung von Convolutional Neural Networks*, University of Hamburg, Germany, Bachelor thesis, 2018

[Hinkle et al. 2003] Hinkle, D.E. ; Wiersma, W. ; Jurs, S.G.: *Applied statistics for the behavioral sciences*. Fifth edition. Houghton Mifflin, 2003

[Hoi et al. 2015] Hoi, S.C. ; Wu, X. ; Liu, H. ; Wu, Y. ; Wang, H. ; Xue, H. ; Wu, Q.: LOGO-Net: Large-scale deep logo detection and brand recognition with deep region-based convolutional networks. *arXiv preprint arXiv:1511.02462*, 2015

[Hoiem et al. 2007] Hoiem, D. ; Efros, A.A. ; Hebert, M.: Recovering surface layout from an image. *International Journal of Computer Vision (IJCV)* 75(1), 2007

[Horbert et al. 2015] Horbert, E. ; Martín García, G. ; Frintrop, S. ; Leibe, B.: Sequence-level object candidates based on saliency for generic object recognition on mobile systems. In: *International Conference on Robotics and Automation (ICRA)*, 2015

[Hosang et al. 2015] Hosang, J. ; Benenson, R. ; Dollár, P. ; Schiele, B.: What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 38(4), 2015

[Hou et al. 2017] Hou, Q. ; Cheng, M.M. ; Hu, X. ; Borji, A. ; Tu, Z. ; Torr, P.H.: Deeply supervised salient object detection with short connections. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Hsu et al. 2020] Hsu, H.K. ; Yao, C.H. ; Tsai, Y.H. ; Hung, W.C. ; Tseng, H.Y. ; Singh, M. ; Yang, M.H.: Progressive domain adaptation for object detection. In: *Winter Conference on Applications of Computer Vision (WACV)*, 2020

[Hu et al. 2017a] Hu, H. ; Lan, S. ; Jiang, Y. ; Cao, Z. ; Sha, F.: FastMask: Segment multi-scale object candidates in one shot. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Hu et al. 2017b] Hu, P. ; Shuai, B. ; Liu, J. ; Wang, G.: Deep level sets for salient object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Hu et al. 2015] Hu, Z. ; Zou, Q. ; Li, Q.: Watershed superpixel. In: *International Conference on Image Processing (ICIP)*, 2015

[Humayun et al. 2014] Humayun, A. ; Li, F. ; Rehg, J.M.: RIGOR: Reusing inference in graph cuts for generating object regions. In: *Computer Vision and Pattern Recognition (CVPR)*, 2014

[Humayun et al. 2015] Humayun, A. ; Li, F. ; Rehg, J.M.: The middle child problem: Revisiting parametric min-cut and seeds for object proposals. In: *International Conference on Computer Vision (ICCV)*, 2015

[Iandola et al. 2015] Iandola, F.N. ; Shen, A. ; Gao, P. ; Keutzer, K.: DeepLogo: Hitting logo recognition with the deep neural network hammer. *arXiv preprint arXiv:1510.02131*, 2015

[Illowsky and Dean 2018] Illowsky, B. ; Dean, S.: *Introductory statistics*. OpenStax, 2018

[Ioffe and Szegedy 2015] Ioffe, S. ; Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning (ICML)*, 2015

[Irving 2016] Irving, B.: maskSLIC: regional superpixel generation with application to local pathology characterisation in medical images. *arXiv preprint arXiv:1606.09518*, 2016

[Isensee and Maier-Hein 2020] Isensee, F. ; Maier-Hein, K.H.: OR-UNet: An optimized robust residual U-Net for instrument segmentation in endoscopic images. *arXiv preprint arXiv:2004.12668*, 2020

[Islam et al. 2021] Islam, M.A. ; Kowal, M. ; Esser, P. ; Jia, S. ; Ommer, B. ; Derpanis, K.G. ; Bruce, N.: Shape or texture: Understanding discriminative features in CNNs. In: *International Conference on Learning Representations (ICLR)*, 2021

[Itti et al. 1998] Itti, L. ; Koch, C. ; Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 20(11), 1998

[Jaderberg et al. 2015] Jaderberg, M. ; Simonyan, K. ; Zisserman, A. ; Kavukcuoglu, K.: Spatial transformer networks. In: *Advances in Neural Information Processing Systems (NeurIPS)*, 2015

[Jampani et al. 2018] Jampani, V. ; Sun, D. ; Liu, M.Y. ; Yang, M.H. ; Kautz, J.: Superpixel sampling networks. In: *European Conference on Computer Vision (ECCV)*, 2018

[Janoch et al. 2013] Janoch, A. ; Karayev, S. ; Jia, Y. ; Barron, J.T. ; Fritz, M. ; Saenko, K. ; Darrell, T.: A category-level 3D object dataset: Putting the Kinect to work. In: *ICCV Workshop on Consumer Depth Cameras for Computer Vision*, 2013

[Johanson 2021] Johanson, R.: *Apple detection using AttentionMask on the MinneApple dataset*, University of Hamburg, Germany, Bachelor thesis, 2021

[Kalantidis et al. 2011] Kalantidis, Y. ; Pueyo, L.G. ; Trevisiol, M. ; van Zwol, R. ; Avrithis, Y.: Scalable triangulation-based logo recognition. In: *International Conference on Multimedia Retrieval (ICMR)*, 2011

[Kanezaki and Harada 2015] Kanezaki, A. ; Harada, T.: 3D Selective Search for obtaining object candidates. In: *International Conference on Intelligent Robots and Systems (IROS)*, 2015

[Kanizsa and Gerbino 1976] Kanizsa, G. ; Gerbino, W.: Convexity and symmetry in figure-ground organization. In: Henle, M. (Ed.): *Vision and Artifact*. Springer, 1976

[Karpathy et al. 2013] Karpathy, A. ; Miller, S. ; Fei-Fei, L.: Object discovery in 3D scenes via shape analysis. In: *International Conference on Robotics and Automation (ICRA)*, 2013

[Ke et al. 2018] Ke, T.W. ; Hwang, J.J. ; Liu, Z. ; Yu, S.X.: Adaptive affinity fields for semantic segmentation. In: *European Conference on Computer Vision (ECCV)*, 2018

[Kim et al. 2022] Kim, D. ; Lin, T.Y. ; Angelova, A. ; Kweon, I.S. ; Kuo, W.: Learning open-world object proposals without learning to classify. *IEEE Robotics and Automation Letters (RA-L)* 7(2), 2022

[Kingma and Ba 2015] Kingma, D.P. ; Ba, J.: Adam: A method for stochastic optimization. In: *International Conference for Learning Representations (ICLR)*, 2015

[Kipf and Welling 2017] Kipf, T.N. ; Welling, M.: Semi-supervised classification with graph convolutional networks. In: *International Conference for Learning Representations (ICLR)*, 2017

[Kirillov et al. 2020] Kirillov, A. ; Wu, Y. ; He, K. ; Girshick, R.: PointRend: Image segmentation as rendering. In: *Computer Vision and Pattern Recognition (CVPR)*, 2020

[Koch and Ullman 1985] Koch, C. ; Ullman, S.: Shifts in selective visual attention: Towards the underlying neural circuitry. *Human Neurobiology* 4(4), 1985

[Koirala et al. 2019a] Koirala, A. ; Walsh, K.B. ; Wang, Z. ; McCarthy, C.: Deep learning – Method overview and review of use for fruit detection and yield estimation. *Computers and Electronics in Agriculture (COMPAG)* 162, 2019

[Koirala et al. 2019b] Koirala, A. ; Walsh, K.B. ; Wang, Z. ; McCarthy, C.: Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'. *Precision Agriculture (PA)* 20(6), 2019

[Kong et al. 2017] Kong, T. ; Sun, F. ; Yao, A. ; Liu, H. ; Lu, M. ; Chen, Y.: RON: Reverse connection with objectness prior networks for object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Kong et al. 2016] Kong, T. ; Yao, A. ; Chen, Y. ; Sun, F.: HyperNet: Towards accurate region proposal generation and joint object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Krähenbühl and Koltun 2014] Krähenbühl, P. ; Koltun, V.: Geodesic object proposals. In: *European Conference on Computer Vision (ECCV)*, 2014

[Krizhevsky et al. 2012] Krizhevsky, A. ; Sutskever, I. ; Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems (NeurIPS)*, 2012

[Kruthiventi et al. 2017] Kruthiventi, S.S. ; Ayush, K. ; Babu, R.V.: DeepFix: A fully convolutional neural network for predicting human eye fixations. *IEEE Transactions on Image Processing (TIP)* 26(9), 2017

[Krähenbühl and Koltun 2015] Krähenbühl, P. ; Koltun, V.: Learning to propose objects. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Kuo et al. 2015] Kuo, W. ; Hariharan, B. ; Malik, J.: DeepBox: Learning objectness with convolutional networks. In: *International Conference on Computer Vision (ICCV)*, 2015

[Kwak et al. 2017] Kwak, S. ; Hong, S. ; Han, B.: Weakly supervised semantic segmentation using superpixel pooling network. In: *AAAI Conference on Artificial Intelligence (AAAI)*, 2017

[Lampert et al. 2009] Lampert, C.H. ; Blaschko, M.B. ; Hofmann, T.: Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 31(12), 2009

[Laradji et al. 2019] Laradji, I.H. ; Vázquez, D. ; Schmidt, M.: Where are the masks: Instance segmentation with image-level supervision. In: *British Machine Vision Conference (BMVC)*, 2019

[Lauri and Frintrop 2017] Lauri, M. ; Frintrop, S.: Object proposal generation applying the distance dependent Chinese restaurant process. In: *Scandinavian Conference on Image Analysis (SCIA)*, 2017

[Lee et al. 2016] Lee, G. ; Tai, Y.W. ; Kim, J.: Deep saliency with encoded low level distance map and high level features. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Lee and Hui 2018] Lee, L.H. ; Hui, P.: Interaction methods for smart glasses: A survey. *IEEE Access* 6, 2018

[Lee et al. 2017] Lee, S.H. ; Jang, W.D. ; Kim, C.S.: Contour-constrained superpixels for image and video processing. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Lee et al. 2014] Lee, T. ; Fidler, S. ; Dickinson, S.: Multi-cue mid-level grouping. In: *Asian Conference on Computer Vision (ACCV)*, 2014

[Lee and Park 2020] Lee, Y. ; Park, J.: CenterMask: Real-time anchor-free instance segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2020

[Levinshtein et al. 2009] Levinshtein, A. ; Stere, A. ; Kutulakos, K.N. ; Fleet, D.J. ; Dickinson, S.J. ; Siddiqi, K.: TurboPixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 31(12), 2009

[Li and Yu 2016] Li, G. ; Yu, Y.: Deep contrast learning for salient object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Li and Gupta 2018] Li, Y. ; Gupta, A.: Beyond grids: Learning graph representations for visual recognition. In: *Advances in Neural Information Processing Systems (NeurIPS)*, 2018

[Li and Chen 2015] Li, Z. ; Chen, J.: Superpixel segmentation using linear spectral clustering. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Lin et al. 2017a] Lin, G. ; Milan, A. ; Shen, C. ; Reid, I.: RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Lin et al. 2019] Lin, S. ; Qin, F. ; Bly, R.A. ; Moe, K.S. ; Hannaford, B.: Automatic sinus surgery skill assessment based on instrument segmentation and tracking in endoscopic video. In: *MICCAI International Workshop on Multiscale Multimodal Medical Imaging*, 2019

[Lin et al. 2017b] Lin, T.Y. ; Dollár, P. ; Girshick, R. ; He, K. ; Hariharan, B. ; Belongie, S.: Feature pyramid networks for object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Lin et al. 2014] Lin, T.Y. ; Maire, M. ; Belongie, S. ; Hays, J. ; Perona, P. ; Ramanan, D. ; Dollár, P.: Microsoft COCO: Common objects in context. In: *European Conference on Computer Vision (ECCV)*, 2014

[Liu et al. 2015] Liu, F. ; Shen, C. ; Lin, G.: Deep convolutional neural fields for depth estimation from a single image. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Liu et al. 2020] Liu, L. ; Ouyang, W. ; Wang, X. ; Fieguth, P. ; Chen, J. ; Liu, X. ; Pietikäinen, M.: Deep learning for generic object detection: A survey. *International Journal of Computer Vision (IJCV)* 128(2), 2020

[Liu et al. 2011] Liu, M.Y. ; Tuzel, O. ; Ramalingam, S. ; Chellappa, R.: Entropy rate superpixel segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2011

[Liu et al. 2018] Liu, S. ; Qi, L. ; Qin, H. ; Shi, J. ; Jia, J.: Path aggregation network for instance segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2018

[Liu et al. 2016a] Liu, W. ; Anguelov, D. ; Erhan, D. ; Szegedy, C. ; Reed, S. ; Fu, C.Y. ; Berg, A.C.: SSD: Single shot multibox detector. In: *European Conference on Computer Vision (ECCV)*, 2016

[Liu et al. 2016b] Liu, Y.J. ; Yu, C.C. ; Yu, M.J. ; He, Y.: Manifold SLIC: A fast method to compute content-sensitive superpixels. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Long et al. 2015] Long, J. ; Shelhamer, E. ; Darrell, T.: Fully convolutional networks for semantic segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Lowe 2004] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)* 60(2), 2004

[Lu et al. 2015] Lu, C. ; Liu, S. ; Jia, J. ; Tang, C.K.: Contour Box: Rejecting object proposals without explicit closed contours. In: *International Conference on Computer Vision (ICCV)*, 2015

[Lu et al. 2018] Lu, H.F. ; Du, X. ; Chang, P.L.: Toward scale-invariance and position-sensitive region proposal networks. In: *European Conference on Computer Vision (ECCV)*, 2018

[Lu et al. 2020] Lu, X. ; Wang, W. ; Danelljan, M. ; Zhou, T. ; Shen, J. ; Van Gool, L.: Video object segmentation with episodic graph memory networks. In: *European Conference on Computer Vision (ECCV)*, 2020

[Lu et al. 2019] Lu, Y. ; Chen, Y. ; Zhao, D. ; Chen, J.: Graph-FCN for image semantic segmentation. In: *International Symposium on Neural Networks (ISNN)*, 2019

[Luengo et al. 2016] Luengo, I. ; Basham, M. ; French, A.P.: SMURFS: Superpixels from multi-scale refinement of super-regions. In: *British Machine Vision Conference (BMVC)*, 2016

[Ma et al. 2017] Ma, J. ; Ming, A. ; Huang, Z. ; Wang, X. ; Zhou, Y.: Object-level proposals. In: *International Conference on Computer Vision (ICCV)*, 2017

[MacKay-Brandt 2011] MacKay-Brandt, A.: Focused attention. In: Kreutzer, J.S. (Ed.) ; DeLuca, J. (Ed.) ; Caplan, B. (Ed.): *Encyclopedia of Clinical Neuropsychology*. Springer, 2011

[Mahadevan and Vasconcelos 2012] Mahadevan, V. ; Vasconcelos, N.: Biologically inspired object tracking using center-surround saliency mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 35(3), 2012

[Maier-Hein et al. 2021] Maier-Hein, L. ; Wagner, M. ; Ross, T. ; Reinke, A. ; Bodenstedt, S. ; Full, P.M. ; Hempe, H. ; Mindroc-Filimon, D. ; Scholz, P. ; Tran, T.N. ; Bruno, P. ; Kisilenko, A. ; Müller, B. ; Davitashvili, T. ; Capek, M. ; Tizabi, M.D. ; Eisenmann, M. ; Adler, T.J. ; Gröhl, J. ; Schellenberg, M. ; Seidlitz, S. ; Lai, T.Y.E. ; Pekdemir, B. ; Roethlingshoefer, V. ; Both, F. ; Bittel, S. ; Mengler, M. ; Mündermann, L. ; Apitz, M. ; Kopp-Schneider, A. ; Speidel, S. ; Nickel, F. ; Probst, P. ; Kenngott, H.G. ; Müller-Stich, B.P.: Heidelberg colorectal data set for surgical data science in the sensor operating room. *Scientific Data* 8(1), 2021

[Malisiewicz and Efros 2007] Malisiewicz, T. ; Efros, A.A.: Improving spatial support for objects via multiple segmentations. In: *British Machine Vision Conference (BMVC)*, 2007

[Manén et al. 2013] Manén, S. ; Guillaumin, M. ; Van Gool, L.: Prime object proposals with randomized Prim's algorithm. In: *International Conference on Computer Vision (ICCV)*, 2013

[Maninis et al. 2016] Maninis, K.K. ; Pont-Tuset, J. ; Arbeláez, P. ; Van Gool, L.: Convolutional oriented boundaries. In: *European Conference on Computer Vision (ECCV)*, 2016

[Maninis et al. 2017] Maninis, K.K. ; Pont-Tuset, J. ; Arbeláez, P. ; Van Gool, L.: Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 40(4), 2017

[Marchesotti et al. 2009] Marchesotti, L. ; Cifarelli, C. ; Csurka, G.: A framework for visual saliency detection with applications to image thumbnailing. In: *International Conference on Computer Vision (ICCV)*, 2009

[Martin et al. 2004] Martin, D.R. ; Fowlkes, C.C. ; Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 26(5), 2004

[Martin et al. 2001] Martin, D.R. ; Fowlkes, C.C. ; Tal, D. ; Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: *International Conference on Computer Vision (ICCV)*, 2001

[Martín García et al. 2015] Martín García, G. ; Potapova, E. ; Werner, T. ; Zillich, M. ; Vincze, M. ; Frintrop, S.: Saliency-based object discovery on RGB-D data with a late-fusion approach. In: *International Conference on Robotics and Automation (ICRA)*, 2015

[Meyer 1994] Meyer, F.: Topographic distance and watershed lines. *Signal Processing* 38(1), 1994

[Milner and Heywood 1989] Milner, A.D. ; Heywood, C.A.: A disorder of lightness discrimination in a case of visual form agnosia. *Cortex* 25(3), 1989

[Moore et al. 2010] Moore, A.P. ; Prince, S.J. ; Warrell, J.: "Lattice Cut" - Constructing superpixels using layer constraints. In: *Computer Vision and Pattern Recognition (CVPR)*, 2010

[Moore et al. 2008] Moore, A.P. ; Prince, S.J. ; Warrell, J. ; Mohammed, U. ; Jones, G.: Superpixel lattices. In: *Computer Vision and Pattern Recognition (CVPR)*, 2008

[Mostajabi et al. 2015] Mostajabi, M. ; Yadollahpour, P. ; Shakhnarovich, G.: Feedforward semantic segmentation with zoom-out features. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Neubert and Protzel 2012] Neubert, P. ; Protzel, P.: Superpixel benchmark and comparison. In: *Forum Bildverarbeitung*, 2012

[Neubert and Protzel 2014] Neubert, P. ; Protzel, P.: Compact watershed and preemptive SLIC: On improving trade-offs of superpixel segmentation algorithms. In: *International Conference on Pattern Recognition (ICPR)*, 2014

[Nikolov et al. 2018] Nikolov, S. ; Blackwell, S. ; Zverovitch, A. ; Mendes, R. ; Livne, M. ; De Fauw, J. ; Patel, Y. ; Meyer, C. ; Askham, H. ; Romera-Paredes, B. ; Kelly, C. ; Karthikesalingam, A. ; Chu, C. ; Carnell, D. ; Boon, C. ; D'Souza, D. ; Moinuddin, S.A. ; Garie, B. ; McQuinlan, Y. ; Ireland, S. ; Hampton, K. ; Fuller, K. ; Montgomery, H. ; Rees, G. ; Mustafa Suleyman1, T.B. ; Hughes, C.O. ; Ledsam, J.R. ; Ronneberger, O.: Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy. *arXiv preprint arXiv:1809.04430*, 2018

[Novotny and Matas 2015] Novotny, D. ; Matas, J.: Cascaded sparse spatial bins for efficient and effective generic object detection. In: *International Conference on Computer Vision (ICCV)*, 2015

[Ošep et al. 2018] Ošep, A. ; Mehner, W. ; Voigtlaender, P. ; Leibe, B.: Track, then decide: Category-agnostic vision-based multi-object tracking. In: *International Conference on Robotics and Automation (ICRA)*, 2018

[Ošep et al. 2020] Ošep, A. ; Voigtlaender, P. ; Weber, M. ; Luiten, J. ; Leibe, B.: 4D generic video object proposals. In: *International Conference on Robotics and Automation (ICRA)*, 2020

[Pakhomov et al. 2019] Pakhomov, D. ; Premachandran, V. ; Allan, M. ; Azizian, M. ; Navab, N.: Deep residual learning for instrument segmentation in robotic surgery. In: *MICCAI International Workshop on Multiscale Multimodal Medical Imaging*, 2019

[Papon et al. 2013] Papon, J. ; Abramov, A. ; Schoeler, M. ; Worgotter, F.: Voxel cloud connectivity segmentation-supervoxels for point clouds. In: *Computer Vision and Pattern Recognition (CVPR)*, 2013

[Park et al. 2017] Park, H. ; Jeong, J. ; Yoo, Y. ; Kwak, N.: Superpixel-based semantic segmentation trained by statistical process control. In: *British Machine Vision Conference (BMVC)*, 2017

[Pashler 1997] Pashler, H.: *The Psychology of Attention*. MIT Press, 1997

[Peña-Barragán et al. 2011] Peña-Barragán, J.M. ; Ngugi, M.K. ; Plant, R.E. ; Six, J.: Object-based crop identification using multiple vegetation indices, textural features and crop phenology. *Remote Sensing of Environment (RSE)* 115(6), 2011

[Perazzi et al. 2017] Perazzi, F. ; Khoreva, A. ; Benenson, R. ; Schiele, B. ; Sorkine-Hornung, A.: Learning video object segmentation from static images. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Perazzi et al. 2012] Perazzi, F. ; Krähenbühl, P. ; Pritch, Y. ; Hornung, A.: Saliency filters: Contrast based filtering for salient region detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2012

[Pierce and Karlin 1957] Pierce, J.R. ; Karlin, J.E.: Reading rates and the information rate of a human channel. *Bell System Technical Journal (BSTJ)* 36(2), 1957

[Pinheiro et al. 2015] Pinheiro, P.O. ; Collobert, R. ; Dollár, P.: Learning to segment object candidates. In: *Advances in Neural Information Processing Systems (NeurIPS)*, 2015

[Pinheiro et al. 2016] Pinheiro, P.O. ; Lin, T.Y. ; Collobert, R. ; Dollár, P.: Learning to refine object segments. In: *European Conference on Computer Vision (ECCV)*, 2016

[Plummer et al. 2018] Plummer, B.A. ; Kordas, P. ; Kiapour, M.H. ; Zheng, S. ; Piramuthu, R. ; Lazebnik, S.: Conditional image-text embedding networks. In: *European Conference on Computer Vision (ECCV)*, 2018

[Pont-Tuset et al. 2017] Pont-Tuset, J. ; Arbeláez, P. ; Barron, J.T. ; Marques, F. ; Malik, J.: Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39(1), 2017

[Qiao et al. 2017] Qiao, S. ; Shen, W. ; Qiu, W. ; Liu, C. ; Yuille, A.: ScaleNet: Guiding object proposal generation in supermarkets and beyond. In: *International Conference on Computer Vision (ICCV)*, 2017

[Qin et al. 2019] Qin, X. ; Zhang, Z. ; Huang, C. ; Gao, C. ; Dehghan, M. ; Jagersand, M.: BASNet: Boundary-aware salient object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Quintero et al. 2018] Quintero, C.P. ; Li, S. ; Pan, M.K. ; Chan, W.P. ; Van der Loos, H.M. ; Croft, E.: Robot programming through augmented trajectories in augmented reality. In: *International Conference on Intelligent Robots and Systems (IROS)*, 2018

[Rahtu et al. 2011] Rahtu, E. ; Kannala, J. ; Blaschko, M.: Learning a category independent object detection cascade. In: *International Conference on Computer Vision (ICCV)*, 2011

[Rajalingham et al. 2018] Rajalingham, R. ; Issa, E.B. ; Bashivan, P. ; Kar, K. ; Schmidt, K. ; DiCarlo, J.J.: Large-scale, high-resolution comparison of the core visual object recognition behavior of humans, monkeys, and state-of-the-art deep artificial neural networks. *Journal of Neuroscience* 38(33), 2018

[Rantalankila et al. 2014] Rantalankila, P. ; Kannala, J. ; Rahtu, E.: Generating object segmentation proposals using global and local search. In: *Computer Vision and Pattern Recognition (CVPR)*, 2014

[Redmon et al. 2016] Redmon, J. ; Divvala, S. ; Girshick, R. ; Farhadi, A.: You only look once: Unified, real-time object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Redmon and Farhadi 2018] Redmon, J. ; Farhadi, A.: YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018

[Ren et al. 2016] Ren, S. ; He, K. ; Girshick, R. ; Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39(6), 2016

[Ren and Malik 2003] Ren, X. ; Malik, J.: Learning a classification model for segmentation. In: *International Conference on Computer Vision (ICCV)*, 2003

[Rensink 2000] Rensink, R.A.: The dynamic representation of scenes. *Visual Cognition* 7(1-3), 2000

[Rensink and Enns 1995] Rensink, R.A. ; Enns, J.T.: Preemption effects in visual search: Evidence for low-level grouping. *Psychological Review* 102(1), 1995

[Rohrbach et al. 2016] Rohrbach, A. ; Rohrbach, M. ; Hu, R. ; Darrell, T. ; Schiele, B.: Grounding of textual phrases in images by reconstruction. In: *European Conference on Computer Vision (ECCV)*, 2016

[Romberg et al. 2011] Romberg, S. ; Pueyo, L.G. ; Lienhart, R. ; van Zwol, R.: Scalable logo recognition in real-world images. In: *International Conference on Multimedia Retrieval (ICMR)*, 2011

[RoyChowdhury et al. 2019] RoyChowdhury, A. ; Chakrabarty, P. ; Singh, A. ; Jin, S. ; Jiang, H. ; Cao, L. ; Learned-Miller, E.: Automatic adaptation of object detectors to new domains using self-training. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Roß et al. 2021] Ross, T. ; Reinke, A. ; Full, P.M. ; Wagner, M. ; Kenngott, H. ; Apitz, M. ; Hempe, H. ; Mindroc-Filimon, D. ; Scholz, P. ; Tran, T.N. ; Bruno, P. ; Arbeláez, P. ; Bian, G.B. ; Bodenstedt, S. ; Bolmgren, J.L. ; Bravo-Sánchez, L. ; Chen, H.B. ; González, C. ; Guo, D. ; Halvorsen, P. ; Heng, P.A. ; Hosgor, E. ; Hou, Z.G. ; Isensee, F. ; Jha, D. ; Jiang, T. ; Jin, Y. ; Kirtac, K. ; Kletz, S. ; Leger, S. ; Li, Z. ; Maier-Hein, K.H. ; Ni, Z.L. ; Riegler, M.A. ; Schoeffmann, K. ; Shi, R. ; Speidel, S. ; Stenzel, M. ; Twick, I. ; Wang, G. ; Wang, J. ; Wang, L. ; Wang, L. ; Zhang, Y. ; Zhou, Y.J. ; Zhu, L. ; Wiesenfarth, M. ; Kopp-Schneider, A. ; Müller-Stich, B.P. ; Maier-Hein, L.: Comparative validation of multi-instance instrument segmentation in endoscopy: Results of the ROBUST-MIS 2019 challenge. *Medical Image Analysis (MedIA)* 70, 2021

[Russakovsky et al. 2015] Russakovsky, O. ; Deng, J. ; Su, H. ; Krause, J. ; Satheesh, S. ; Ma, S. ; Huang, Z. ; Karpathy, A. ; Khosla, A. ; Bernstein, M. ; Berg, A.C. ; Fei-Fei, L.: ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)* 115(3), 2015

[Russell et al. 2006] Russell, B.C. ; Freeman, W.T. ; Efros, A.A. ; Sivic, J. ; Zisserman, A.: Using multiple segmentations to discover objects and their extent in image collections. In: *Computer Vision and Pattern Recognition (CVPR)*, 2006

[Russell et al. 2008] Russell, B.C. ; Torralba, A. ; Murphy, K.P. ; Freeman, W.T.: LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)* 77(1), 2008

[Sa et al. 2016] Sa, I. ; Ge, Z. ; Dayoub, F. ; Upcroft, B. ; Perez, T. ; McCool, C.: DeepFruits: A fruit detection system using deep neural networks. *Sensors* 16(8), 2016

[Sadeghi 2019] Sadeghi, M.A.: *Lokalisierung von Flugzeug-Leitwerken*, University of Hamburg, Germany, Bachelor thesis, 2019

[Saito et al. 2021] Saito, K. ; Hu, P. ; Darrell, T. ; Saenko, K.: Learning to detect every thing in an open world. *arXiv preprint arXiv:2112.01698*, 2021

[Saito et al. 2019] Saito, K. ; Ushiku, Y. ; Harada, T. ; Saenko, K.: Strong-weak distribution alignment for adaptive object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Salton and McGill 1986] Salton, G. ; McGill, M.J.: *Introduction to modern information retrieval*. McGraw-Hill, 1986

[Schwartz et al. 2009] Schwartz, W.R. ; Kembhavi, A. ; Harwood, D. ; Davis, L.S.: Human detection using partial least squares analysis. In: *International Conference on Computer Vision (ICCV)*, 2009

[Sheikh et al. 2007] Sheikh, Y.A. ; Khan, E.A. ; Kanade, T.: Mode-seeking by medoidshifts. In: *International Conference on Computer Vision (ICCV)*, 2007

[Shen et al. 2014] Shen, J. ; Du, Y. ; Wang, W. ; Li, X.: Lazy random walks for superpixel segmentation. *IEEE Transactions on Image Processing (TIP)* 23(4), 2014

[Shen et al. 2016] Shen, J. ; Hao, X. ; Liang, Z. ; Liu, Y. ; Wang, W. ; Shao, L.: Real-time superpixel segmentation by DBSCAN clustering algorithm. *IEEE Transactions on Image Processing (TIP)* 25(12), 2016

[Shi and Malik 2000] Shi, J. ; Malik, J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 22(8), 2000

[Silberman et al. 2012] Silberman, N. ; Hoiem, D. ; Kohli, P. ; Fergus, R.: Indoor segmentation and support inference from RGBD images. In: *European Conference on Computer Vision (ECCV)*, 2012

[Simonyan and Zisserman 2015] Simonyan, K. ; Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations (ICLR)*, 2015

[Song et al. 2015] Song, S. ; Lichtenberg, S.P. ; Xiao, J.: SUN RGB-D: A RGB-D scene understanding benchmark suite. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Sonka et al. 2014] Sonka, M. ; Hlavac, V. ; Boyle, R.: *Image processing, analysis, and machine vision*. Fourth edition. Cengage Learning, 2014

[Srivastava et al. 2015] Srivastava, R.K. ; Greff, K. ; Schmidhuber, J.: Highway networks. *arXiv preprint arXiv:1505.00387*, 2015

[Stutz et al. 2018] Stutz, D. ; Hermans, A. ; Leibe, B.: Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding (CVIU)* 166, 2018

[Su et al. 2017a] Su, H. ; Gong, S. ; Zhu, X.: WebLogo-2M: Scalable logo detection by deep learning from the web. In: *ICCV Workshop on Web-Scale Vision and Social Media*, 2017

[Su et al. 2017b] Su, H. ; Zhu, X. ; Gong, S.: Deep learning logo detection with data expansion by synthesising context. In: *Winter Conference on Applications of Computer Vision (WACV)*, 2017

[Sun and Batra 2015] Sun, Q. ; Batra, D.: SubmodBoxes: Near-optimal search for a set of diverse object proposals. In: *Advances in Neural Information Processing Systems (NeurIPS)*, 2015

[Sünderhauf et al. 2018] Sünderhauf, N. ; Brock, O. ; Scheirer, W. ; Hadsell, R. ; Fox, D. ; Leitner, J. ; Upcroft, B. ; Abbeel, P. ; Burgard, W. ; Milford, M. ; Corke, P.: The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research (IJRR)* 37(4-5), 2018

[Suzuki et al. 2018] Suzuki, T. ; Akizuki, S. ; Kato, N. ; Aoki, Y.: Superpixel convolution for segmentation. In: *International Conference on Image Processing (ICIP)*, 2018

[Szegedy et al. 2016] Szegedy, C. ; Vanhoucke, V. ; Ioffe, S. ; Shlens, J. ; Wojna, Z.: Rethinking the inception architecture for computer vision. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Tang et al. 2018] Tang, P. ; Wang, X. ; Wang, A. ; Yan, Y. ; Liu, W. ; Huang, J. ; Yuille, A.: Weakly supervised region proposal network and object detection. In: *European Conference on Computer Vision (ECCV)*, 2018

[Tang and Wu 2016] Tang, Y. ; Wu, X.: Saliency detection via combining region-level and pixel-level predictions with CNNs. In: *European Conference on Computer Vision (ECCV)*, 2016

[Tepper et al. 2017] Tepper, O.M. ; Rudy, H.L. ; Lefkowitz, A. ; Weimer, K.A. ; Marks, S.M. ; Stern, C.S. ; Garfein, E.S.: Mixed reality with HoloLens: Where virtual reality meets augmented reality in the operating room. *Plastic and reconstructive surgery (PRS)* 140(5), 2017

[Tomasi and Manduchi 1998] Tomasi, C. ; Manduchi, R.: Bilateral filtering for gray and color images. In: *International Conference on Computer Vision (ICCV)*, 1998

[Toyama et al. 2012] Toyama, T. ; Kieninger, T. ; Shafait, F. ; Dengel, A.: Gaze guided object recognition using a head-mounted eye tracker. In: *Symposium on Eye Tracking Research and Applications (ETRA)*, 2012

[Treisman and Gelade 1980] Treisman, A.M. ; Gelade, G.: A feature-integration theory of attention. *Cognitive Psychology* 12(1), 1980

[Tu et al. 2018] Tu, W.C. ; Liu, M.Y. ; Jampani, V. ; Sun, D. ; Chien, S.Y. ; Yang, M.H. ; Kautz, J.: Learning superpixels with segmentation-aware affinity loss. In: *Computer Vision and Pattern Recognition (CVPR)*, 2018

[Tüzkö et al. 2018] Tüzkö, A. ; Herrmann, C. ; Manger, D. ; Beyerer, J.: Open set logo detection and retrieval. In: *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2018

[Uijlings et al. 2013] Uijlings, J.R. ; van de Sande, K.E. ; Gevers, T. ; Smeulders, A.W.: Selective search for object recognition. *International Journal of Computer Vision (IJCV)* 104(2), 2013

[van de Sande et al. 2011] van de Sande, K.E. ; Uijlings, J.R. ; Gevers, T. ; Smeulders, A.W.: Segmentation as selective search for object recognition. In: *International Conference on Computer Vision (ICCV)*, 2011

[Van den Bergh et al. 2012] Van den Bergh, M. ; Boix, X. ; Roig, G. ; de Capitani, B. ; Van Gool, L.: SEEDS: Superpixels extracted via energy-driven sampling. In: *European Conference on Computer Vision (ECCV)*, 2012

[Van den Bergh et al. 2015] Van den Bergh, M. ; Boix, X. ; Roig, G. ; Van Gool, L.: SEEDS: Superpixels extracted via energy-driven sampling. *International Journal of Computer Vision (IJCV)* 111(3), 2015

[Vedaldi and Soatto 2008] Vedaldi, A. ; Soatto, S.: Quick shift and kernel methods for mode seeking. In: *European Conference on Computer Vision (ECCV)*, 2008

[Veksler et al. 2010] Veksler, O. ; Boykov, Y. ; Mehrani, P.: Superpixels and supervoxels in an energy optimization framework. In: *European Conference on Computer Vision (ECCV)*, 2010

[Vicente et al. 2011] Vicente, S. ; Rother, C. ; Kolmogorov, V.: Object cosegmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2011

[Vincent and Soille 1991] Vincent, L. ; Soille, P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 13(6), 1991

[von Hofsten and Spelke 1985] von Hofsten, C. ; Spelke, E.S.: Object perception and object-directed reaching in infancy. *Journal of Experimental Psychology: General* 114(2), 1985

[Wang et al. 2015] Wang, C. ; Zhao, L. ; Liang, S. ; Zhang, L. ; Jia, J. ; Wei, Y.: Object proposal by multi-branch hierarchical segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Wang et al. 2019] Wang, J. ; Chen, K. ; Yang, S. ; Loy, C.C. ; Lin, D.: Region proposal by guided anchoring. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Wang and Wang 2012] Wang, J. ; Wang, X.: VCells: Simple and efficient superpixels using edge-weighted centroidal Voronoi tessellations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 34(6), 2012

[Wang et al. 2013a] Wang, P. ; Zeng, G. ; Gan, R. ; Wang, J. ; Zha, H.: Structure-sensitive superpixels via geodesic distance. *International Journal of Computer Vision (IJCV)* 103(1), 2013

[Wang et al. 2013b] Wang, Q. ; Nuske, S. ; Bergerman, M. ; Singh, S.: Automated crop yield estimation for apple orchards. In: *International Symposium on Experimental Robotics (ISER)*, 2013

[Wang et al. 2021a] Wang, W. ; Feiszli, M. ; Wang, H. ; Tran, D.: Unidentified video objects: A benchmark for dense, open-world segmentation. In: *International Conference on Computer Vision (ICCV)*, 2021

[Wang et al. 2021b] Wang, W. ; Xie, E. ; Li, X. ; Fan, D.P. ; Song, K. ; Liang, D. ; Lu, T. ; Luo, P. ; Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: *International Conference on Computer Vision (ICCV)*, 2021

[Wang et al. 2021c] Wang, Y. ; Wei, Y. ; Qian, X. ; Zhu, L. ; Yang, Y.: AINet: Association implantation for superpixel segmentation. In: *International Conference on Computer Vision (ICCV)*, 2021

[Weikersdorfer et al. 2013] Weikersdorfer, D. ; Schick, A. ; Cremers, D.: Depth-adaptive supervoxels for RGB-D video segmentation. In: *International Conference on Image Processing (ICIP)*, 2013

[Werner et al. 2015] Werner, T. ; Martín García, G. ; Frintrop, S.: Saliency-guided object candidates based on Gestalt principles. In: *International Conference on Computer Vision Systems (ICVS)*, 2015

[Wertheimer 1922] Wertheimer, M.: Untersuchungen zur Lehre von der Gestalt. *Psychologische Forschung* 1(1), 1922

[Wilms and Frintrop 2017] Wilms, C. ; Frintrop, S.: Edge adaptive seeding for superpixel segmentation. In: *German Conference on Pattern Recognition (GCPR)*, 2017

[Wilms and Frintrop 2018] Wilms, C. ; Frintrop, S.: AttentionMask: Attentive, efficient object proposal generation focusing on small objects. In: *Asian Conference on Computer Vision (ACCV)*, 2018

[Wilms and Frintrop 2020] Wilms, C. ; Frintrop, S.: Superpixel-based refinement for object proposal generation. In: *International Conference on Pattern Recognition (ICPR)*, 2020

[Wilms and Frintrop 2021] Wilms, C. ; Frintrop, S.: DeepFH segmentations for superpixel-based object proposal refinement. *Image and Vision Computing (IMAVIS)* 114, 2021

[Wilms et al. 2022a] Wilms, C. ; Gerlach, A.M. ; Schmitz, R. ; Frintrop, S.: Segmenting medical instruments in minimally invasive surgeries using AttentionMask. *arXiv preprint arXiv:2203.11358*, 2022

[Wilms et al. 2020] Wilms, C. ; Heid, R. ; Sadeghi, M.A. ; Ribbrock, A. ; Frintrop, S.: Which airline is this? Airline logo detection in real-world weather conditions. In: *International Conference on Pattern Recognition (ICPR)*, 2020

[Wilms et al. 2022b] Wilms, C. ; Johanson, R. ; Frintrop, S.: Localizing small apples in complex apple orchard environments. *arXiv preprint arXiv:2202.11372*, 2022

[Wolfe et al. 1989] Wolfe, J.M. ; Cave, K.R. ; Franzel, S.L.: Guided search: An alternative to the feature integration model for visual search. *Journal of Experimental Psychology: Human Perception and Performance* 15(3), 1989

[Wu et al. 2013] Wu, Y. ; Lim, J. ; Yang, M.H.: Online object tracking: A benchmark. In: *Computer Vision and Pattern Recognition (CVPR)*, 2013

[Wu and He 2018] Wu, Y. ; He, K.: Group normalization. In: *European Conference on Computer Vision (ECCV)*, 2018

[Xiao et al. 2013] Xiao, J. ; Owens, A. ; Torralba, A.: SUN3D: A database of big spaces reconstructed using SfM and object labels. In: *International Conference on Computer Vision (ICCV)*, 2013

[Xiao et al. 2015] Xiao, Y. ; Lu, C. ; Tsougenis, E. ; Lu, Y. ; Tang, C.K.: Complexity-adaptive distance metric for object proposals generation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Xie et al. 2021] Xie, C. ; Xiang, Y. ; Mousavian, A. ; Fox, D.: Unseen object instance segmentation for robotic environments. *IEEE Transactions on Robotics (T-RO)* 37(5), 2021

[Xie et al. 2020] Xie, E. ; Sun, P. ; Song, X. ; Wang, W. ; Liu, X. ; Liang, D. ; Shen, C. ; Luo, P.: PolarMask: Single shot instance segmentation with polar representation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2020

[Xie and Tu 2015] Xie, S. ; Tu, Z.: Holistically-nested edge detection. In: *International Conference on Computer Vision (ICCV)*, 2015

[Xu et al. 2015] Xu, K. ; Ba, J. ; Kiros, R. ; Cho, K. ; Courville, A. ; Salakhudinov, R. ; Zemel, R. ; Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: *International Conference on Machine Learning (ICML)*, 2015

[Yamaguchi et al. 2012] Yamaguchi, K. ; Kiapour, M.H. ; Ortiz, L.E. ; Berg, T.L.: Parsing clothing in fashion photographs. In: *Computer Vision and Pattern Recognition (CVPR)*, 2012

[Yang et al. 2020] Yang, F. ; Sun, Q. ; Jin, H. ; Zhou, Z.: Superpixel segmentation with fully convolutional networks. In: *Computer Vision and Pattern Recognition (CVPR)*, 2020

[Yang et al. 2016] Yang, Z. ; He, X. ; Gao, J. ; Deng, L. ; Smola, A.: Stacked attention networks for image question answering. In: *Computer Vision and Pattern Recognition (CVPR)*, 2016

[Yanulevskaya et al. 2014] Yanulevskaya, V. ; Uijlings, J. ; Sebe, N.: Learning to group objects. In: *Computer Vision and Pattern Recognition (CVPR)*, 2014

[Yao et al. 2015] Yao, J. ; Boben, M. ; Fidler, S. ; Urtasun, R.: Real-time coarse-to-fine topologically preserving segmentation. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Yu and Koltun 2016] Yu, F. ; Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: *International Conference on Learning Representations (ICLR)*, 2016

[Yu et al. 2017] Yu, F. ; Koltun, V. ; Funkhouser, T.: Dilated residual networks. In: *Computer Vision and Pattern Recognition (CVPR)*, 2017

[Yu et al. 2019] Yu, Y. ; Zhang, K. ; Yang, L. ; Zhang, D.: Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. *Computers and Electronics in Agriculture (COMPAG)* 163, 2019

[Zhai et al. 2021] Zhai, Q. ; Li, X. ; Yang, F. ; Chen, C. ; Cheng, H. ; Fan, D.P.: Mutual graph learning for camouflaged object detection. In: *Computer Vision and Pattern Recognition (CVPR)*, 2021

[Zhang et al. 2019a] Zhang, C. ; Lin, G. ; Liu, F. ; Yao, R. ; Shen, C.: CANet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Zhang et al. 2021] Zhang, J. ; Aviles-Rivero, A.I. ; Heydecker, D. ; Zhuang, X. ; Chan, R. ; Schönlieb, C.B.: Dynamic spectral residual superpixels. *Pattern Recognition* 112, 2021

[Zhang and Gao 2020] Zhang, J. ; Gao, X.: Object extraction via deep learning-based marker-free tracking framework of surgical instruments for laparoscope-holder robots. *International Journal of Computer Assisted Radiology and Surgery (IJCARS)* 15(8), 2020

[Zhang et al. 2019b] Zhang, L. ; Li, X. ; Arnab, A. ; Yang, K. ; Tong, Y. ; Torr, P.H.: Dual graph convolutional network for semantic segmentation. In: *British Machine Vision Conference (BMVC)*, 2019

[Zhang et al. 2014] Zhang, N. ; Donahue, J. ; Girshick, R. ; Darrell, T.: Part-based R-CNNs for fine-grained category detection. In: *European Conference on Computer Vision (ECCV)*, 2014

[Zhang et al. 2011a] Zhang, Y. ; Hartley, R. ; Mashford, J. ; Burn, S.: Superpixels via pseudo-boolean optimization. In: *International Conference on Computer Vision (ICCV)*, 2011

[Zhang et al. 2011b] Zhang, Z. ; Warrell, J. ; Torr, P.H.: Proposal generation for object detection using cascaded ranking SVMs. In: *Computer Vision and Pattern Recognition (CVPR)*, 2011

[Zhao et al. 2014] Zhao, Q. ; Liu, Z. ; Yin, B.: Cracking BING and beyond. In: *British Machine Vision Conference (BMVC)*, 2014

[Zhao et al. 2015] Zhao, R. ; Ouyang, W. ; Li, H. ; Wang, X.: Saliency detection by multi-context deep learning. In: *Computer Vision and Pattern Recognition (CVPR)*, 2015

[Zhao et al. 2017] Zhao, W. ; Jiao, L. ; Ma, W. ; Zhao, J. ; Zhao, J. ; Liu, H. ; Cao, X. ; Yang, S.: Superpixel-based multiple local CNN for panchromatic and multispectral image classification. *IEEE Transactions on Geoscience and Remote Sensing (TGRS)* 55(7), 2017

[Zhaoping 2014] Zhaoping, L.: *Understanding vision: Theory, models, and data.* Oxford University Press, 2014

[Zheng et al. 2017] Zheng, L. ; Yang, Y. ; Tian, Q.: SIFT meets CNN: A decade survey of instance retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 40(5), 2017

[Zhu et al. 2021] Zhu, L. ; She, Q. ; Zhang, B. ; Lu, Y. ; Lu, Z. ; Li, D. ; Hu, J.: Learning the superpixel in a non-iterative and lifelong manner. In: *Computer Vision and Pattern Recognition (CVPR)*, 2021

[Zhu et al. 2019] Zhu, X. ; Pang, J. ; Yang, C. ; Shi, J. ; Lin, D.: Adapting object detectors via selective cross-domain alignment. In: *Computer Vision and Pattern Recognition (CVPR)*, 2019

[Zitnick and Dollár 2014] Zitnick, C.L. ; Dollár, P.: Edge Boxes: Locating object proposals from edges. In: *European Conference on Computer Vision (ECCV)*, 2014

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den

Christian Wilms