Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Disentanglement, Compositionality, Specification: Representation Learning with Generative Adversarial Networks

Dissertation

submitted to the Universität Hamburg, Faculty of Mathematics, Informatics and Natural Sciences, Department of Informatics, in partial fulfillment of the requirements for the degree of *Doctor rerum naturalium* (Dr. rer. nat.)

Tobias Hinz

Hamburg, 2021

Date of Submission:
8th February 2021

Date of Oral Defense:
28th April 2021

Dissertation Committee:

Prof. Dr. Simone Frintrop
Department of Computer Science
Universität Hamburg, Germany

Prof. Dr. Stefan Wermter (advisor)
Department of Computer Science
Universität Hamburg, Germany

Prof. Dr. Jianwei Zhang (chair)
Department of Computer Science
Universität Hamburg, Germany

Abstract

Modern technology allows for access to large amounts of data. However, this data is only useful if we can structure it, extract information, and learn connections and relationships between different data points. To achieve this, it is essential to represent the data in a way that facilitates organization and knowledge extraction. Depending on what we want to do with the data, different characteristics of the chosen representation method may be more or less important. In this thesis, we develop approaches for learning image representations that possess certain desirable characteristics: disentanglement, compositionality, and specification.

Disentanglement implies that the representation should model factors of variation of the underlying data generating process. Learning these factors will allow the representation to model different data points in a concise manner and potentially allows us to learn novel interrelations and dependencies. Compositionality postulates that representations are constructed from independent concepts in a hierarchical manner. Finally, specification means that we can focus our learning capabilities on a single task or goal, allowing us to spend the full representation capacity to model individual and complex objects in isolation.

Our first approach introduces an approach utilizing a Generative Adversarial Network (GAN) to learn disentangled representations from visual data. Our model learns meaningful and interpretable concepts, e.g. object classes and characteristics, on its own without supervision. Adding a small amount of supervision allows for more control over the learned representations while still allowing the model to learn unlabeled concepts from the data itself. In our second approach, we learn compositional representations that allow a GAN to model complex visual scenes consisting of multiple different objects. Experiments show that the model generalizes to several novel settings such as more or fewer objects, novel object positions and sizes, and novel object-attribute combinations. We also identify several shortcomings of current evaluation methods for these kinds of visual scenes and introduce a novel evaluation metric that correlates well with human perception. Finally, we show how we can use GANs to learn object-specific representations from only a few data points. By making use of several implicit biases and specific data augmentation methods we can learn good representations from only a single example. We can learn even better representations if we have slightly more (e.g. 15) training examples available. The learned representations are, by design, highly specific to the given object but only need very limited training data.

The models we introduce in this thesis each address one specific representation characteristic that we believe is useful for many different tasks. These representations can not only be used for knowledge discovery but also allow for a more structured approach to modeling complex environments or complex objects without access to large data sets. We highlight the connections between all three of our approaches and highlight several directions for future research. Specifically, we identify different ways to combine our approaches such that the resulting representations benefit from the advantages of the individual characteristics.

Zusammenfassung

Moderne Technologien ermöglichen den Zugriff auf große Datenmengen. Diese sind jedoch nur nützlich, wenn daraus Strukturen und Beziehungen gelernt werden können. Um dies zu erreichen, müssen die Daten auf eine Weise repräsentiert werden, die die Wissensextraktion erleichtert. Je nach Anwendungsfeld können verschiedene Eigenschaften einer Repräsentationsmethode mehr oder weniger wichtig sein. Wir entwickeln Ansätze zum Lernen von Bildrepräsentationen, die bestimmte vorteilhafte Eigenschaften besitzen: Entflechtung, Kompositionalität und Spezifikation.

Entflechtung bedeuted, dass die Repräsentation die Faktoren des zugrunde liegenden Datenerzeugungsprozesses modellieren sollte. Das Erlernen dieser Faktoren ermöglicht es der Repräsentation, verschiedene Datenpunkte auf prägnante Weise zu modellieren und erlaubt es neuartige Zusammenhänge und Abhängigkeiten zu erlernen. Kompositionalität postuliert, dass Repräsentationen aus unabhängigen Konzepten in einer hierarchischen Weise konstruiert werden. Spezifikation bedeutet, dass wir unsere Lernfähigkeiten unseres Modells auf ein einziges Ziel fokussieren können, was es uns ermöglicht, die volle Repräsentationskapazität für die Modellierung einzelner komplexer Objekte zu verwenden.

Unser erstes Modell ist ein "Generative Adversarial Network" (GAN) das entflochtene Repräsentationen von visuellen Daten lernt. Unser Modell lernt sinnvolle und interpretierbare Konzepte, z.Bsp. Objektklassen und -eigenschaften ohne Überwachung. Das Hinzufügen einer kleinen Menge an Überwachung ermöglicht eine bessere Kontrolle über die gelernten Repräsentationen. In unserem zweiten Ansatz lernen wir kompositionelle Repräsentationen, die es einem GAN erlauben, komplexe visuelle Szenen zu modellieren, die aus mehreren verschiedenen Objekten bestehen. Experimente zeigen, dass das Modell zu verschiedenen neuartigen Kombinationen, wie neuartigen Objektpositionen und -größen sowie neuartigen Objekt-Attribut-Kombinationen, generalisiert. Wir identifizieren mehrere Schwachstellen aktueller Bewertungsmethoden für diese Art von visuellen Szenen und stellen eine neue Metrik vor, die stark mit der menschlichen Wahrnehmung korreliert. Schließlich zeigen wir, wie wir GANs verwenden können, um objektspezifische Repräsentationen aus nur wenigen Datenpunkten zu lernen. Indem wir das Modell durch einen Einfluss-induzierenden Bias unterschiedlich ausrichten und duch spezifische Methoden zur Datenerweiterung können wir gute Repräsentationen von wenigen oder einem einzigen Beispiel lernen. Die gelernten Repräsentationen sind hochspezifisch für das jeweilige Objekt, benötigen aber nur sehr wenige Trainingsdaten.

Unsere entwickelten Modelle adressieren jeweils eine spezifische Eigenschaft von Repräsentationen, von der wir glauben, dass sie für viele verschiedene Aufgaben nützlich ist. Diese Repräsentationen können nicht nur zur Wissensentdeckung verwendet werden, sondern ermöglichen auch einen strukturierteren Ansatz zur Modellierung komplexer Umgebungen oder Objekte ohne Zugang zu großen Datensätzen. Wir zeigen Verbindungen zwischen unseren drei Ansätzen und zeigen mehrere Richtungen für zukünftige Forschung auf. Insbesondere präsentieren wir verschiedene Möglichkeiten, unsere Ansätze so zu kombinieren, dass die resultierenden Repräsentationen von den Vorteilen der einzelnen Merkmale profitieren.

Contents

Chapter 1

Introduction

Imagine being asked to calculate LXXXVII - XLI? What if you were asked to calculate V×CL? Most people would convert these numbers to the Arabic numerals first and only then calculate $87-41$ or 5×150 respectively. While these calculations are simple, more complex calculations might require manual (written) calculations which are much easier to do with Arabic numerals. If an automatic calculator is used, chances are that it would be better to represent the calculations as 0101 0111 - 0010 1001 and 0000 0101× 1001 0110 respectively, as most calculators work with a binary representation. Clearly, some representations for numbers are better than others for some tasks.

Other examples also highlight the importance of choosing good representations. The current Latin alphabet which is widely used consists of comparatively few "simple" symbols. These symbols on their own are meaningless, but by combining these symbols in the right order we can obtain words that represent real objects or abstract concepts. Other writing systems, e.g. Egyptian hieroglyphs, on the other hand, have individual symbols for real-world objects and concepts. As a result, the Latin alphabet is much smaller than other alphabets but can still represent the same (or even more) concepts.

All these examples show how important it is to have good representations for a given concept. The representation's characteristics directly affect how well subsequent tasks can be solved. Different representations might also vary in size, in their interpretability and editability, their ease of use, and many other characteristics. While the definition of a "good" representation is task dependent, Bengio et al. [2013] have identified a number of desirable characteristics that learned representations should possess. In this thesis, we examine three of these desirable characteristics – disentanglement, compositionality, and specificity – and propose approaches for how we can learn representations that possess these characteristics.

Disentanglement means that different parts of the representation encode different data generating factors, making the representation interpretable and controllable. For example, a typical human's face could be represented by various characteristics (eye color, hair color, shape of the nose, etc.) and a disentangled representation would represent each of these characteristics individually. *Compositionality* implies that the representation is made up of several different parts that encode distinct

parts of the environment, e.g. individual objects. This helps with modeling complex environments and allows generalization to novel layouts and scenes. For example, given an image depicting a group of people we could model each face's characteristics individually and use the combination of these individual representations to represent the full image. Finally, *specificity* describes representations that are specifically learned for a single concept, e.g. a single object or scene. Through this, we can learn detailed representations of complex objects and structures from very little data and with little previous knowledge. For example, we could learn a representation specifically for one individual face to model different characteristics, e.g. emotions, in great detail and specifically tailored to that face.

We choose to focus on these three characteristics as they positively affect various learning and modeling processes. Furthermore, each of these characteristics directly addresses weaknesses or challenges in the other two characteristics. While we do not examine this, it is also possible to combine these characteristics to learn even more powerful representations. For example, a compositional representation could be constructed from several disentangled representations, which, in turn, could be learned specifically for different kinds of objects. As such, while this thesis develops and evaluates these three properties in isolation, we identify several promising directions through which these representation characteristics can be combined in future research directions.

1.1 Research Questions

Generative models are powerful models with more capabilities than e.g. discriminative models. In contrast to discriminative approaches, generative models can learn representations without any available labels in a completely unsupervised way. While this makes them potentially useful for many domains, learning good representations without labels is difficult and, to a degree, ill-defined as the quality of a given representation is difficult to evaluate without a task. To address this, there are a number of inductive biases and qualities that are generally considered to be useful and "good" for representations, independent of any downstream tasks [Bengio et al., 2013]. This thesis addresses the question of how we can learn representations that possess some of these desirable characteristics and whether these properties can be helpful for different tasks. We study Generative Adversarial Networks (GANs) as our basic generative model and examine the following questions:

Disentangled representations (Chapter 3): How can we learn disentangled representations with GANs? We examine what kinds of inductive biases and architectural designs we need to learn disentangled representations. We also evaluate how many labels – if any – are necessary for this and what kinds of underlying factors we can learn in this way. Finally, we look into how interpretable and meaningful the learned representations are and how we can evaluate them.

Compositional representations (Chapter 4): Can GANs learn compositional representations that decompose the underlying distribution?

We study how we can learn representations with GANs that allow us to explicitly control various aspects of image generation in complex environments by structuring the environment as a composition of individual objects. We examine whether these representations generalize to novel settings or distributions and how we can evaluate these representations in a quantitative, automatic manner.

Object-specific representations (Chapter 5): Can we use GANs to learn representations specifically for a single object from limited data? We investigate how much data is necessary to learn a representation of a single given object and how we can reduce the amount of necessary data and still obtain good representations.

1.2 Contributions of this Thesis

The focus of this work is on studying Generative Adversarial Nets (GANs) for learning good representations of images. The main contributions of this work are:

Disentangled representations (Chapter 3): We develop a framework for learning disentangled representations with GANs based on no or very few labeled data points. Our novel GAN architecture can learn disentangled representations for both inference at test time and image generation and translation. The model is able to learn data generating factors without any provided labels but can also be "steered" to learn specific factors based on very few labeled data points (e.g. only 100 labeled images for the MNIST data set). For inference, the learned representations can be used to extract data that possesses specific characteristics or features (e.g. specific foreground objects, background colors, contrast, ...) that were learned during training. Additionally, we can control the image generation process based on the disentangled factors and can use the inference and generation capabilities jointly to translate a given image where we only change specific characteristics (e.g. the class label) while keeping everything else unchanged (e.g. the background).

Compositional representations (Chapter 4): We intoduce a novel architecture and evaluation metric for learning compositional representations with GANs. Learning compositional representations with GANs enables us to generate complex scenes with explicit control over object identities, locations, and sizes. To evaluate these representations we develop a new evaluation metric for generative text-to-image synthesis models that evaluates both text-image alignment and image quality in a compositional manner. User studies show that our metric closely reflects the ranking obtained from humans whereas other popular scores predict a different ranking.

Object-specific representations (Chapter 5): We show that we can learn useful representations for a single given object from only very little training data without any prior or external knowledge. Based on only a single training data point we can learn representations that can be used for tasks

such as image generation, image harmonization, image animation, and more. If we have more information about the given object (e.g. 15 data points) we show how we can learn even better representations that allow us to perform tasks such as character reposing and animation.

1.3 Thesis Outline

This thesis is split into six main parts, consisting of an introduction (Chapter 1) and background (Chapter 2) chapter, three chapters that introduce our main approaches to representation learning (Chapter 3, Chapter 4, and Chapter 5), and a conclusion (Chapter 6) in which we discuss our approaches and point to future research directions. Chapter 3 (disentangled representations), Chapter 4 (compositional representations), and Chapter 5 (object-specific representations) all follow the same structure. We first give some general information about the chapter's representations' characteristics we are looking into as well as necessary information about the chapter's background and related work. This is followed by two sections in which we introduce our specific approaches and their respective results. After this, we discuss the advantages and disadvantages of our approaches and critically evaluate them in the context of other approaches. We conclude each of these three chapters with a short section about overall results and trends as well as possible future work in the specific area of the given chapter.

Chapter 2

Background

In this chapter, we will introduce the general framework of Generative Adversarial Networks (GANs) and how they can be used for representation learning. We will first describe what representation learning is and what the different approaches to representation learning are. Following this, we describe what GANs are and how the unconditional variant differs from the conditional one.

2.1 Representation Learning

Many tasks can be very easy or very hard depending on how data is represented [Goodfellow et al., 2016] and, therefore, having good representations of data is essential for efficient use of this data. This is not only true for representations in neural networks, but also in daily life. For example, calculating with numbers in Arabic numeral representations is much easier for humans than doing calculations with binary representations, but the same is not necessarily true for computers. Other examples are many operations in computer science (e.g. accessing or inserting elements) which can be more or less efficient depending on the used representation (e.g. an array or a linked list). Representation learning generally describes different approaches and methodologies to learn "good" representations, where the meaning of "good" and the form of the learned representations are usually defined by a given use case.

When deep neural networks are used to process data they implicitly learn representations of the data across their different layers. What kind of representation is learned usually depends on the final task. If the task is, e.g., image classification, then the last hidden layer in a neural network tries to learn a representation that makes the different classes linearly separable. Depending on the task and the way a neural network is trained will affect the characteristics of the learned representations and what they can be used for. Bengio et al. [2013] introduce several prior representation properties that are generally considered to be "good" regardless of the task. For data points x and y and their respective representations $f(x)$ and $f(y)$ these properties include, among others:

- Smoothness: if $x \approx y$ then that implies $f(x) \approx f(y)$. While this prior is

present in many models it is not clear how well modern deep learning models implement this [Szegedy et al., 2014].

- Hierarchical organization: concepts in the real world can be described by other concepts, where more abstract concepts are higher in the hierarchy and are made up of more basic concepts. This prior is explicitly implemented in deep learning models.
- Manifolds: even if the input data is high-dimensional, the probability mass of many natural distributions concentrates around smaller-dimensional regions [Lu et al., 1998; Vasconcelos and Lippman, 2005]. This holds true for natural images and is exploited in many algorithms such as auto-encoders and other manifold inspired algorithms.
- Temporal and spatial coherence: observations that are close to each other either in time or space tend to be correlated and often result in only a small move along the manifold distribution [Becker and Hinton, 1992].
- Sparsity: given input x, only a small part of the data generating factors are usually relevant. This can be implemented by a representation that is mostly zero and only non-zero for relevant factors x [Olshausen and Field, 1996].

However, depending on the task at hand, many other advantageous properties may exist.

Representation learning approaches can usually be clustered into supervised and unsupervised approaches. Supervised approaches provide a label for each input and the goal is to learn a mapping from the input data to the provided labels (e.g. classification and regression). In unsupervised approaches no labels are available and the goal is to learn a representation of the data that could be useful for downstream tasks (e.g. clustering and outlier prediction). Semi-supervised learning is a mixture of these two approaches where we have labels for some of the data but also data for which no labels are available. Recently, an approach called self-supervised learning applies traditional supervised learning approaches to unlabeled data by extracting labels from the data "for free", usually by withholding part of the data and training the network to predict the missing part. Our models for using GANs for representation learning make use of supervised [Hinz et al., 2019, 2021b,a], semi-supervised [Hinz and Wermter, 2018a], and unsupervised training [Hinz and Wermter, 2018b; Hinz et al., 2021b]. For completeness, we also include a brief section about self-supervised training in this chapter.

2.1.1 Supervised Representation Learning

Supervised representation learning usually deals with solving a specific task, e.g. image classification. During training, we have access to labels for each training instance and train the network to predict the correct label(s) for each training data. For many applications, these labels may be difficult to obtain automatically and, thus, need to be obtained manually through human labor.

When the task is well-defined and enough labeled data is available supervised learning algorithms typically lead to better results than unsupervised learning algorithms. However, obtaining enough labels can be costly and time-consuming.

This is especially the case when expert knowledge is needed to obtain labels (e.g. in the medical domain). As such, algorithms that can learn useful properties of data even without labels have gained in popularity in the last years, especially since a lot of data is available "for free" on the internet.

2.1.2 Unsupervised Representation Learning

Unsupervised representation learning deals with data for which no labels are available. The goal is to learn properties about the data distribution that might be useful for downstream tasks such as clustering or density estimation. One of the simplest approaches to unsupervised representation learning is through dimensionality reduction, i.e. to embed the input data into a lower-dimensional representation. E.g., by training a neural network to reconstruct the input data from this representation the network learns to remove redundancies from the input data and only encode "abstract" information [Kramer, 1991; Hinton and Zemel, 1993; Vincent et al., 2010]. These learned representations can be used as input for subsequent tasks such as classification and clustering.

More recently, several approaches explore unsupervised representation learning via mutual information maximization. The mutual dependence between two random variables x and y quantifies how much information can be gained about one of the variables by observing the other one and measures (informally) the difference between the product of the two marginal distributions $P(x)$ and $P(y)$ and the joint distribution $P(x, y)$. By maximizing the mutual information between different views of the same data point we hope to increase the amount of "information" encoded in the representation. Several approaches have examined this, both with GANs [Chen et al., 2016; Hinz and Wermter, 2018b] and for traditional representation learning evaluated on downstream tasks (usually classification) [Hjelm et al., 2019; Bachman et al., 2019; Tschannen et al., 2020].

2.1.3 Semi-Supervised Representation Learning

Semi-supervised learning is a mixture of supervised and unsupervised learning. In this setting, we usually have a small amount of labeled data and a large amount of unlabeled data available. The goal is to make use of the unlabeled data to increase the final performance on the task defined by the labeled data. Discriminative semi-supervised approaches can mostly be classified as graph-based, entropy-based, consistency-based, or rely on co-training. Graph-based approaches [Zhu and Ghahramani, 2002; Weston et al., 2012] work under the assumption that similar points in the data space (based on some predefined distance metric) should have the same label. Entropy-based approaches [Grandvalet and Bengio, 2005; Rosenberg et al., 2005] encourage the model to make very confident predictions about all data points, regardless of whether labels are available or not. For the supervised training the prediction should be the same as the label, while for unlabeled data points the model is simply trained to have low entropy in its prediction (i.e. it does not matter which class it predicts for a given unlabeled data point, as long as it

is sure about its prediction). Consistency-based approaches [Belkin et al., 2006; Bachman et al., 2014] work on the assumption that small perturbations to the input (e.g. adding a small amount of noise to the image) should not change the model's prediction. Co-training [Blum and Mitchell, 1998; Zhou and Li, 2005] assumes that we have multiple views of the same data which contain enough information on their own to classify the input correctly. These models can be combined with generative models to potentially further increase the final performance [Kingma et al., 2014; Odena, 2016; Denton et al., 2016; Paige et al., 2017].

2.1.4 Self-Supervised Representation Learning

Self-supervised learning is a variant of supervised learning in which the labels are not obtained manually. Broadly speaking, there are two ways in how these models are usually trained. One approach is to withhold some part of the data and train the model to predict the missing data, e.g. by training a network to predict a missing word in a sentence [Mikolov et al., 2013] or to add color to a black-and-white image [Zhang et al., 2016]. The second approach transforms the data somehow and trains the network to identify the applied transformation, e.g. by training a model to identify the correct order of sentences [Lan et al., 2020] or by rotating an image and training the network to identify the angle around which the image was rotated [Gidaris et al., 2018]. Both approaches define a proxy loss through which the network is trained in the hope that the network learns some semantic representation along the way.

2.2 Generative Models

Generative versus Discriminative Models When talking about machine learning models we can broadly distinguish between discriminative and generative models. The main difference between these two kinds of models lies in the kind of distribution they learn. Given some input data $x \in \mathcal{X}$ (e.g. images) and labels $y \in \mathcal{Y}$ (e.g. class labels), discriminative models learn a conditional distribution $P(y|x)$. Generative models, on the other hand, learn either a joint distribution $P(x, y)$ over the input data and labels or, in the absence of labels, learn a distribution $P(x)$ of the data itself.

As a consequence, discriminative models learn a mapping from the input data x to the output data y. Since the model learns a conditional distribution, the full model capacity can be used to learn a decision boundary to correctly classify or regress y given the input x. Because of this, discriminative models tend to perform better than generative models at tasks such as classification and regression.

Learning a generative model is generally considered to be more difficult than learning a discriminative model since the model capacity can not solely be dedicated to learning a decision boundary. While discriminative models only learn this boundary between data samples, generative models generally try to learn how the data is distributed throughout the data space. When labels are available,

generative models can, in theory, perform the same tasks as discriminative models since $P(y|x) = P(x, y)/P(x)$. The advantage of generative models is that they offer capabilities that discriminative models to do not possess, such as generating new data from the data distribution, learning joint probabilities, and offering a natural way of learning something about the data structure without available labels. As such, generative models can be applied to tasks such as unsupervised learning, clustering, dimensionality reduction, density estimation, and much more.

Explicit versus Implicit Generative Models Generative models can be further divided into explicit and implicit models [Goodfellow, 2016]. Explicit models define an explicit density function $P(x)$ of the distribution. These models can be trained via maximum likelihood estimation, i.e. for a model with parameters θ the model's parameters can be optimized such that they maximize the likelihood of the observed training data. Explicit models can be evaluated by calculating the likelihood they assign to a held-out test set and many explicit models learn a data representation that can be used for down-stream tasks such as classification or clustering.

The main challenges here lie in defining a model that can capture the complexity of the training data while still being computationally tractable to optimize. To achieve this, models are either constructed in a way that guarantees a computational tractable density (usually at the cost of parameter efficiency [Dinh et al., 2015, 2017] or sampling speed [Van Den Oord et al., 2016; Van den Oord et al., 2016]), or in a way that allows a tractable approximation of the density (usually at the cost of not having an exact likelihood [Ackley et al., 1985; Kingma and Welling, 2014]).

Implicit models, on the other hand, do not define an explicit density function, but instead only allow for sampling from the data distribution: $x' \sim p_{\text{model}}(x)$. These models are usually trained by comparing the samples from the real distribution $x \sim p(x)$ to samples from the model $x' \sim p_{\text{model}}(x)$. They offer fast sampling from the data distribution and the sample quality often has a higher (perceptual) quality compared to explicit models [Blau and Michaeli, 2018]. However, since they do not model a density it is difficult to evaluate implicit models quantitatively [Shmelkov et al., 2018]. GANs are currently one of the most popular implicit generative models and we will go into more detail in the next sections.

2.2.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] are a form of implicit generative model and allow for drawing samples x' from the learned distribution $p_{\text{model}}(x)$. A GAN consists of two neural networks (see Figure 2.1), usually referred to as discriminator D and generator G, which are trained with opposing objectives. The generator takes as input random noise z sampled from a pre-defined distribution (usually a uniform $\mathcal{U}(-1, 1)$ or normal distribution $\mathcal{N}(0, 1)$) and transforms it into a sample x': $G : z \to x'$. The discriminator D takes as input data samples from either the real distribution $x \sim p_{\text{real}}(x)$ or samples generated by the generator $x' \sim p_{\text{generator}}(x)$ and classifies these samples as either real or fake:

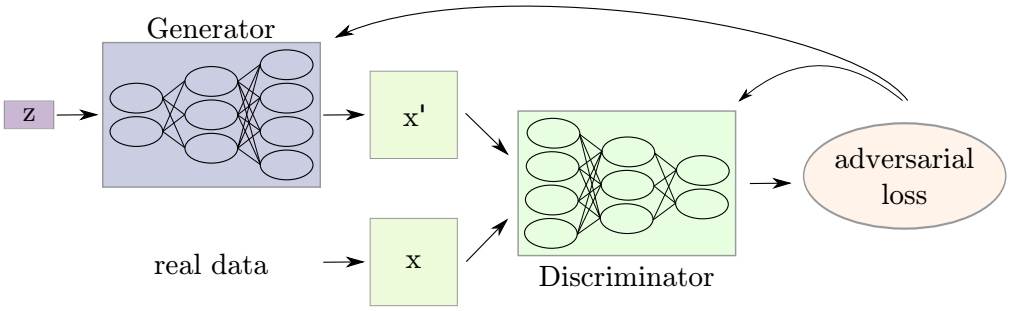


Figure 2.1: Overview of a Generative Adversarial Network (GAN). The generator takes as input a noise vector z and transforms it into a data sample x'. The discriminator takes as input either a real data sample x or a generated sample x' and classifies it as real or generated (adversarial loss). Both the generator and the discriminator are trained with the adversarial loss.

$D : x \to \{0, 1\}$.

The discriminator is trained to classify real images as real and generated images as generated. The discriminator's cost function is:

$$J(D) = -\mathbb{E}_{x \sim p_{\text{data}(x)}} \big[\log D(x)\big] - \mathbb{E}_{z \sim p_z} \big[\log(1 - D(G(z))\big]. \tag{2.1}$$

Since the generator is trained to generate samples that are classified as real by the discriminator a straightforward approach would be to treat the approach as a zero-sum game in which the sum of both networks' cost would be zero: $J(G) = -J(D)$. Since the generator can not affect the discriminator's response to the real data this would result in the following cost function for the generator:

$$J(G) = \mathbb{E}_{z \sim p_z} \big[\log(1 - D(G(z))\big]. \tag{2.2}$$

However, this loss function (Equation 2.2) is not ideal, since the gradients of this cost saturate when the discriminator classifies generated samples as fake with high certainty (see Figure 2.2). In other words, the generator would not get "useful" feedback (based on the gradients) when it generates very bad data but would get "useful" feedback only when it already generates very good data. At the beginning of the training, the generator usually does not produce good data, so it would be more useful to receive good feedback from the discriminator early on in training. To address

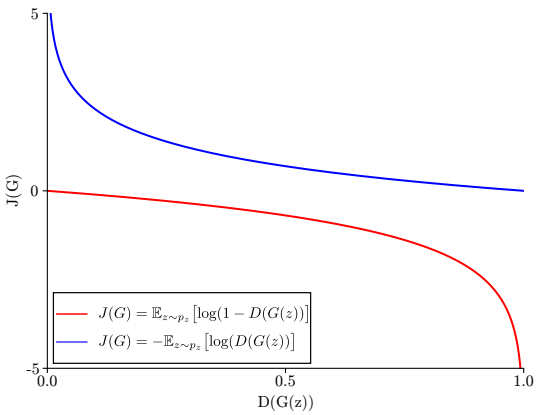


Figure 2.2: When the discriminator classifies inputs as fake with high certainty ($D(G(z)) \approx 0$) the gradients of Equation 2.2 are close to zero, while the gradients of Equation 2.3 provide more information in this case.

this, the most commonly taken ap-
proach is to not flip the sign of the cost function but rather to flip the target of the generator's cost. In Equation 2.2, the generator's goal is to minimize the probability of the discriminator's prediction being correct (on the generated data). Instead, we now train the generator to maximize the probability of the discriminator making wrong predictions on the generated data. The generator's cost function then becomes:

$$J(G) = -\mathbb{E}_{z\sim p_z}\big[\log(D(G(z))\big]. \tag{2.3}$$

Note that this loss function is mostly heuristically motivated with the motivation of providing strong gradients to whichever network is "losing" at a given moment. The two networks are then trained alternately, i.e. while optimizing the parameters of one of the two networks the other network's parameters stay fixed.

Challenges in Training GANs In the computer vision setting, GANs are often employed to generate new images from a given distribution (e.g. faces, birds, or buildings). While GANs show impressive results for image generation [Brock et al., 2019], they still suffer from several drawbacks. Training GANs is relatively unstable and highly dependent on the chosen hyperparameters [Arjovsky and Bottou, 2017; Mescheder et al., 2018]. If the discriminator becomes too strong, the resulting gradients will not provide enough signal for the generator to learn. On the other hand, if the discriminator is too weak, the generator might not get enough feedback to learn to generate good samples. Several approaches such as adding noise to the discriminator [Arjovsky and Bottou, 2017], penalizing discriminator weights [Roth et al., 2017], updated losses [Arjovsky et al., 2017], gradient penalties [Gulrajani et al., 2017], and normalization approaches [Miyato et al., 2018] try to improve the training stability.

Additionally, GANs sometimes suffer from mode collapse, which means that the generator may learn to produce only a very small number of different (highly realistic) samples, i.e. maps large parts of z to the same generated sample. Several approaches try to address this, e.g. by preventing the generator from overfitting to a given discriminator's state [Metz et al., 2017], by adding an additional loss [Srivastava et al., 2017] or by forcing the generator explicitly to map different noise vectors z to different parts of the data distribution [Mao et al., 2019b]. Another challenge is mode dropping in which the generator does not learn full support of the data distribution and instead covers only part of it [Arora and Zhang, 2018; Bau et al., 2019]. Approaches to this are usually similar to approaches that try to solve the mode collapse problem [Mroueh et al., 2017; Sinha et al., 2020].

Despite the challenges in training GANs, they show promising results in many different applications such as image generation, image-to-image translation, image editing, super-resolution, animation, inpainting, and many more. Their capability to generate sharp and realistic images makes them the model of choice for many applications in which image quality is of high importance. Outside of generating images they have also shown their applicability to tasks such as semi-supervised learning [Odena, 2016; Donahue et al., 2017; Dumoulin et al., 2017; Li et al., 2017a; Donahue and Simonyan, 2019], video generation [Saito et al., 2017; Tulyakov et al.,

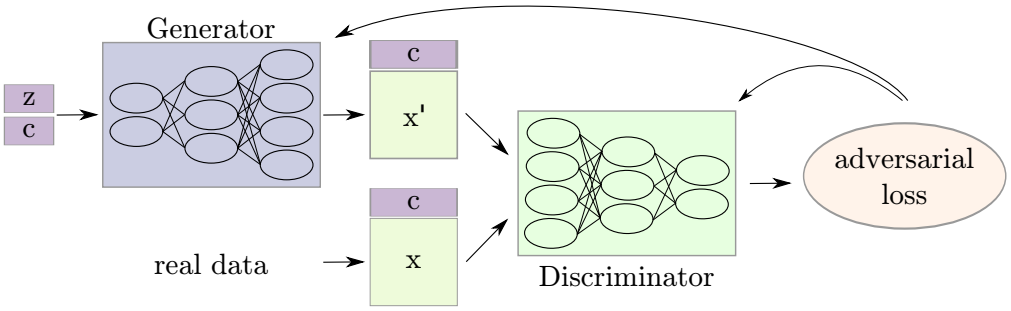


Figure 2.3: Overview of a conditional Generative Adversarial Network (cGAN). Compared to a normal Generative Adversarial Network (Figure 2.1), conditional Generative Adversarial Networks have an additional conditioning variable c (e.g. a class label) which is used as input for both the discriminator and the generator.

2018; Saito et al., 2020], and audio and speech synthesis [Donahue et al., 2019; Bińkowski et al., 2020].

2.2.2 Conditional Generative Adversarial Networks

Mirza and Osindero [2014] introduced the conditional GAN variant in which both the generator and the discriminator are conditioned on additional information (see Figure 2.3). This additional information can take the form of e.g. class labels, textual descriptions, or images. Using this additional information allows applying the GAN framework to tasks such as class-conditional image generation [Mirza and Osindero, 2014], text-to-image synthesis [Reed et al., 2016b], image-to-image translation [Isola et al., 2017; Zhu et al., 2017a], and many more.

In the most basic form [Mirza and Osindero, 2014] the noise vector z and the conditioning information c are simply concatenated and fed as input to the generator: $G : (z, c) \rightarrow x'$. Likewise, the generated or real images and the respective conditioning information c are concatenated and used as input to the discriminator: $D : (x, c) \rightarrow \{0, 1\}$. The used loss terms stay unchanged:

$$J(D) = -\mathbb{E}_{x \sim p_{\text{data}(x)}, c}\big[\log D(x, c)\big] - \mathbb{E}_{z \sim p_z, c}\big[\log(1 - D(G(z, c), c)\big], \tag{2.4}$$

$$J(G) = -\mathbb{E}_{z \sim p_z, c}\big[\log(D(G(z, c), c)\big]. \tag{2.5}$$

When the conditioning information is more complex, e.g. a textual description or an image, the conditioning information is usually embedded into a lower dimensional representation first.

How exactly the conditioning information is used as input for the discriminator and generator is a critical design choice and can affect the final performance. Only concatenating the conditioning information with the input at the very lowest layer for both the discriminator and the generator often does not lead to the best results. Several approaches explore methods to improve the conditioning in GANs. This can, e.g., be achieved by incorporating higher-level interactions [Miyato and Koyama, 2018] or an additional contrastive loss term [Kang and Park, 2020] between learned

representations and the conditioning information in the discriminator. Alternatively, the conditioning information can be used by an additional classifier network [Odena et al., 2017] which is trained to correctly classify the real data and can be used as additional training signal for the generator. Finally, many approaches feed the conditioning information into various layers of both the discriminator and the generator [Zhang et al., 2017b, 2018a; Karras et al., 2019, 2020b] or learn specific features that are used as normalization throughout various layers of the generator [Park et al., 2019].

Chapter 3

Disentangled Representations

As we have seen previously (Section 2.1), there are many characteristics that may be useful for learned representations. Many of these characteristics can be seen as priors that can help to disentangle the underlying factors of variation of a given data distribution. This is made explicit with the concept of disentangled representations [Bengio et al., 2013] which encode different, independent data generating variables in different parts of the learned representations. Disentanglement can make the learned representation more interpretable, easier to modify, and has the potential to make other downstream tasks easier to solve [Morcos et al., 2018]. We will first give an overview of what disentangled representations are and how they can be used and evaluated. Following this, we will show two approaches to learning and using disentangled representations with GANs. We conclude with a critical discussion of our approaches and how they are related to recent developments in the broader community.

3.1 Disentangled Representations

A common assumption is that a data point, which is sampled from a distribution, is encoded by several latent variables that describe the properties of this data point. These latent variables encode the various data generating factors that define all data points in the distribution. Consider, for example, the MNIST data set [LeCun et al., 1998] which consists of images depicting a single digit on an uniform background. Simply put, the latent variables for this distribution could be the *class* $(0-9)$, *position* (xy-coordinate), *rotation* (angle), and *width* (in pixels) of a given digit. Knowing the values for each of these variables would enable us to visualize the given image.

Definition of Disentangled Representations An intuitive definition of disentangled representations is that they should encode each latent data generating factor in a way that makes it independent of the other latent variables, e.g. by representing each latent variable in a disjoint part of the representation. We assume that an observation x is generated from some latent factors z_i via $p(x|z_i)$ and a disentangled representation would represent these individual latent factors $\{z_i\}$

in a disjoint and interpretable manner [Do and Tran, 2020]. For example, the representation of a given MNIST digit's class should not be influenced by its position and width. However, finding a concrete definition of disentangled representations is difficult [Higgins et al., 2018]. This is partly because outside of synthetic data sets it is often not even clear what constitutes a data generating factor. The common desideratum is, therefore, to try and learn a model that disentangles as many data generating factors as possible while discarding as little information as possible in the process.

Higgins et al. [2017] define disentanglement through independence, i.e. two latent variables are disentangled if they are mutually independent. However, this definition is quite restrictive and ignores many realistic settings in which intuitively disentangled characteristics of objects are still strongly correlated (e.g. gender and facial hair) [Träuble et al., 2020]. To address this, Higgins et al. [2018] propose a more general definition of disentanglement in which changing one specific disentangled variable only affects a specific property in the generated data point, while leaving all other properties unchanged. While this definition removes the requirement of mutual independence it is very abstract which limits its applicability across different domains and evaluation approaches. Suter et al. [2019] examine disentanglement from the view of it being the property of a causal process, i.e. the different data generating variables do not affect each other (but may still be correlated). Do and Tran [2020] define disentanglement through the concepts of informativeness (the mutual information between a data point and its representation), separability (different parts of the representation share no information about the data), and interpretability (by providing some predefined concepts). Overall, however, many approaches do not follow a rigid definition of disentanglement, but rather define representations as disentangled if they are interpretable [Chen et al., 2016; Sepliarskaia et al., 2019].

Evaluation of Disentangled Representations Since there is no clear definition of what constitutes a disentangled representation it is challenging to evaluate disentangled representations across different domains and models. As a consequence, visual inspection and interpretability are often the standard metrics [Bengio et al., 2013; Chen et al., 2016; Eastwood and Williams, 2018]. Several evaluation metrics exist for datasets in which the ground truth information about the data generating factors is available [Higgins et al., 2017; Eastwood and Williams, 2018]. However, these metrics are limited to synthetic data sets and can not be used for real-world data for which the exact data generating factors are unknown.

Some metrics try to evaluate disentangled representations in the sense that changing one part of the disentangled representation should change one generative factor while being invariant to all other generative factors [Higgins et al., 2017; Kim and Mnih, 2018; Eastwood and Williams, 2018]. Other metrics approach the evaluation from the perspective that a change in a data generating factor should only affect a specific part of the disentangled representation [Kumar et al., 2018; Chen et al., 2018]. Sepliarskaia et al. [2019] introduce a metric that combines both previous approaches. Other evaluation metrics focus on invariance [Goodfellow

et al., 2009; Cohen and Welling, 2015] or equivariance [Lenc and Vedaldi, 2015]. Do and Tran [2020] provide a more detailed overview of recent evaluation metrics. Note, however, that many (especially older) evaluation metrics do not correlate strongly with perceptual (qualitative) disentanglement [Abdi et al., 2019]. In the following two sections we introduce our approaches for learning and evaluating disentangled representations with GANs.

3.2 Unsupervised Learning of Disentangled Representations

This section presents our work *Inferencing based on unsupervised learning of disentangled representations* by Tobias Hinz and Stefan Wermter published in 2018 at the *European Symposium on Artificial Neural Networks (pp. 61 – 66).*

3.2.1 Introduction

Learning meaningful representations of data is an important step for models to understand the world [Bengio et al., 2013]. Recently, the Generative Adversarial Network (GAN) [Goodfellow et al., 2014] has been proposed as a method that can learn characteristics of data distributions without the need for labels. GANs traditionally consist of a generator G, which generates data from randomly sampled vectors Z, and a discriminator D, which tries to distinguish generated data from real data x. During training, the generator learns to generate realistic data samples $G(Z)$, while the discriminator becomes better at distinguishing between the generated and the real data x. As a result, both the generator and the discriminator learn characteristics about the underlying data distribution without the need for any labels [Radford et al., 2016]. One desirable characteristic of learned representations is disentanglement [Bengio et al., 2013], which means that different parts of the representation encode different factors of the data-generating distribution. This makes representations more interpretable, easier to modify, and is a useful property for many tasks such as classification, clustering, or image captioning.

To achieve this, Chen et al. [2016] introduced a GAN variant in which the generator's input is split into two parts z and c. Here, z encodes unstructured noise while c encodes meaningful, data-generating factors. Through enforcing high mutual information between c and and the generated images $G(z, c)$ the generator is trained using the inputs c as meaningful encodings for certain image characteristics. For example, a ten-dimensional categorical code for c could represent the ten different digit classes in the MNIST data set. Since no labels are provided the generator has to learn by itself which image characteristics can be represented through c. One drawback of this model is that the only way to perform inference, i.e. map real data samples into a (disentangled) representation, is to use the discriminator. However, there is no guarantee that the discriminator learns good representations of the data in general, as it is trained to discriminate between real and generated data and may therefore focus only on features that are helpful for discriminating these two, but

are not necessarily descriptive of the data distribution in general [Donahue et al., 2017]. Zhang et al. [2017a] tried to enforce disentangled representations in order to improve the controllability of the generator. The latent representation is split up into two parts encoding meaningful information and unknown factors of variation. Two additional inference networks are introduced to enforce the disentanglement between the two parts of the latent representation. While this setup yields a better controllability over the generative process it depends on labeled samples for its training objective and can not discover unknown data-generating factors, but only encodes known factors of variation (obtained through labels) in its disentangled representation.

Donahue et al. [2017] and Dumoulin et al. [2017] introduced an extension which includes an encoder E that learns the encodings of real data samples. The discriminator gets as input both the data sample x (either real or generated) and the according representation (either Z or $E(x)$) and has to classify them as either coming from the generator or the encoder. The generator and the encoder try to fool the discriminator into misclassifying the samples. As a result, the encoder E learns to approximate the inverse of the generator G and can be used to map real data samples into representations for other applications. However, in these approaches the representations follow a simple prior, e.g. a Gaussian or uniform distribution, and do not exhibit any disentangled properties.

Our model, the Bidirectional-InfoGAN, integrates some of these approaches by extending traditional GANs with an encoder that learns disentangled representations in an unsupervised setting. After training, the encoder can map data points to meaningful, disentangled representations which can potentially be used for different tasks such as classification, clustering, or image captioning. Compared to the InfoGAN [Chen et al., 2016] we introduce an encoder to mitigate the problems of using a discriminator for both the adversarial loss and the inference task. Unlike the Structured GAN [Zhang et al., 2017a] our training procedure is completely unsupervised, can detect unknown data-generating factors, and only introduces one additional inference network (the encoder). In contrast to the Bidirectional GAN [Donahue et al., 2017; Dumoulin et al., 2017] we replace the simple prior on the latent representation with a distribution that is amenable to disentangled representations and introduce an additional loss for the encoder and the generator to achieve disentangled representations. On the MNIST, CelebA [Liu et al., 2015], and SVHN [Netzer et al., 2011] data sets we show that the encoder does learn interpretable representations which encode meaningful properties of the data distribution. Using these we can sample images that exhibit certain characteristics, e.g. digit identity and specific stroke widths for the MNIST data set, or different hair colors and clothing accessories in the CelebA data set.

3.2.2 Methodology

Our model, shown in Fig. 3.1, consists of a generator G, a discriminator D, and an encoder E, which are implemented as neural networks. The input vector Z that is given to the generator G is made up of two parts $Z = (z, c)$. Here, z is sampled from

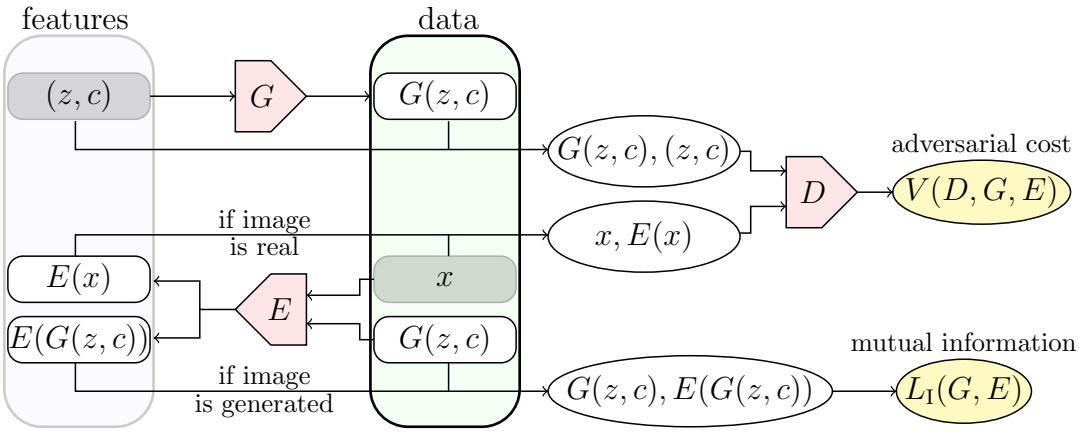


Figure 3.1: High-level overview of the Bidirectional-InfoGAN. The generator G generates images from the vector (z, c) and tries to fool the discriminator into classifying them as real. The encoder E encodes images into a representation and tries to fool the discriminator D into misclassifying them as fake if its input is a real image while trying to approximate $P(c|x)$ if its input is a generated image.

a uniform distribution, $z \sim U(-1, 1)$, and is used to represent unstructured noise in the images. On the other hand, c is the part of the representation that encodes meaningful information in a disentangled manner and is made up of both categorical values c_{cat} and continuous values c_{cont}. G takes Z as input and transforms it into an image x, i.e. $G : Z \to x$.

E is a convolutional network that gets as input either real or fake images and encodes them into a latent representation $E : x \to Z$. D gets as input an image x and the corresponding representation Z concatenated along the channel axis. It then tries to classify the pair as coming either from the generator G or the encoder E, i.e. $D : Z \times x \to \{0, 1\}$, while both G and E try to fool the discriminator into misclassifying its input. As a result the original GAN minimax game [Goodfellow et al., 2014] is extended and becomes:

$$\min_{G,E} \max_{D} V(D, G, E) = \mathbb{E}_{x \sim P_{\text{data}}}[log D(x, E(x))] + \mathbb{E}_{Z \sim P_Z}[log(1 - D(G(Z), Z))],$$

where $V(D, G, E)$ is the adversarial cost as depicted in Fig. 3.1.

In order to force the generator to use the information provided in c we maximize the mutual information I between c and $G(z, c)$. Maximizing the mutual information directly is hard, as it requires the posterior $P(c|x)$ and we therefore follow the approach by Chen et al. [2016] and define an auxiliary distribution $E(c|x)$ to approximate $P(c|x)$. We then maximize the lower bound $L_I(G, E) = \mathbb{E}_{c \sim P(c), z \sim P(z), x \sim G(z,c)}[log\ E(c|x)] + H(c) \leq I(c; G(z, c))$, where $L_I(G, E)$ is the mutual information depicted in Fig. 3.1. For simplicity reasons we fix the distribution over c and, therefore, the entropy term $H(c)$ is treated as a constant. In our case E is the encoder network which gets images generated by G as input and is trained to approximate the unknown posterior $P(c|x)$. For categorical c_{cat}

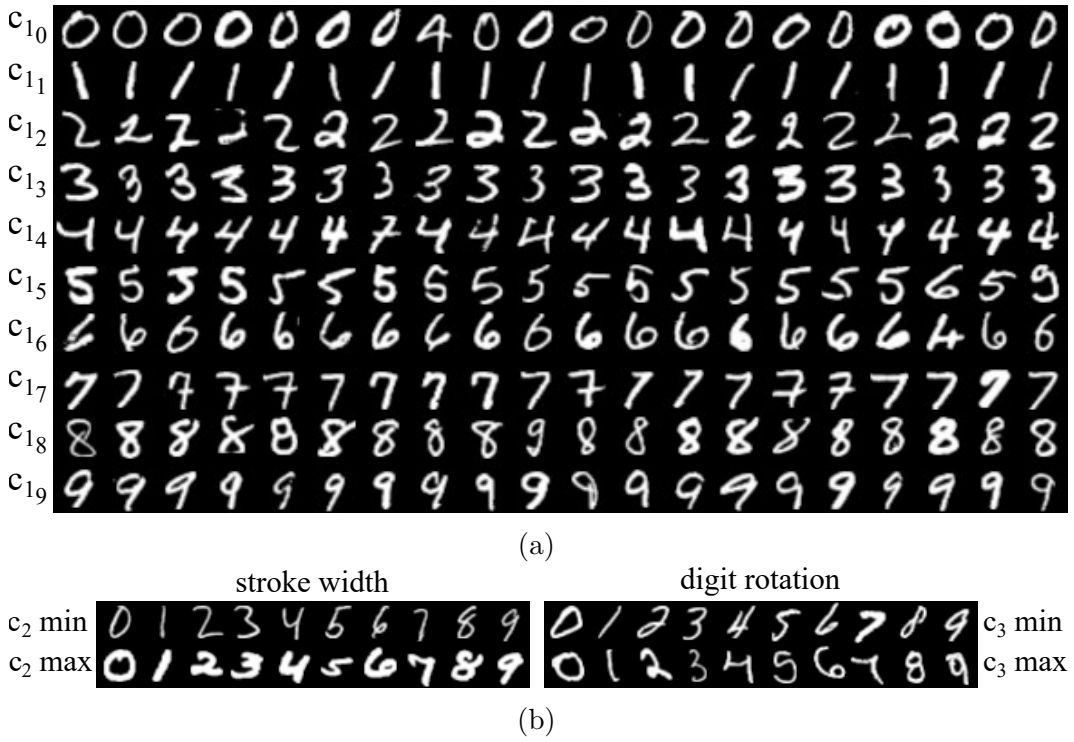


Figure 3.2: Images sampled from the MNIST test set. (a) Each row represents one value of the ten-dimensional code c_1, which encodes different digits despite never seeing labels during the training process. (b) Images with maximum and minimum values for c_2 and c_3 for each categorical value from c_1.

we use the softmax nonlinearity to represent $E(c_{\text{cat}}|x)$ while we treat the posterior $E(c_{\text{cont}}|x)$ of continuous c_{cont} as a factored Gaussian. Given this structure, the minimax game for the Bidirectional-InfoGAN (BInfoGAN) is then

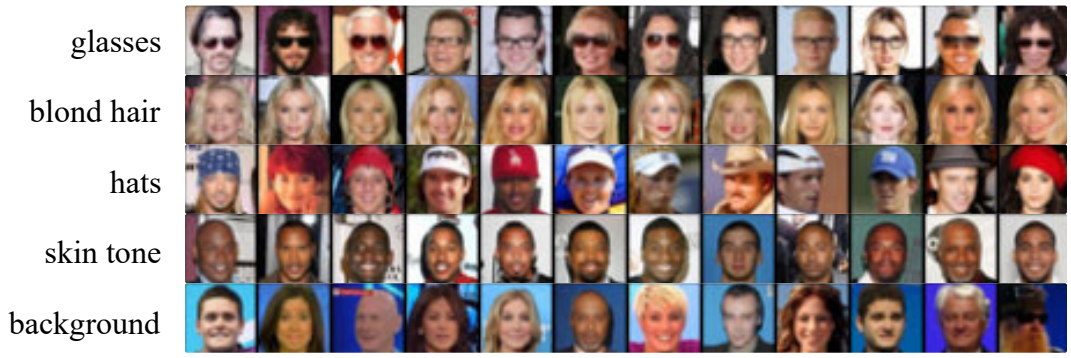
$$\min_{G,E} \max_{D} V_{\text{BInfoGAN}}(D, G, E) = V(D, G, E) - \lambda L_I(G, E)$$

where λ determines the strength of the impact of the mutual information criterion L_I and is set to 1.0 in all our experiments.

3.2.3 Experiments

We perform experiments on the MNIST, the CelebA [Liu et al., 2015], and the SVHN [Netzer et al., 2011] data set. While the final performance of the model is likely influenced by choosing the "optimal" characteristics for c this is usually not possible, since we do not know all data-generating factors beforehand. When choosing the characteristics and dimensionality of the disentangled vector c we therefore mostly stick with the values previously chosen by Chen et al. [2016].

On the MNIST data set we model the latent code c with one categorical variable $c_1 \sim Cat(K = 10, p = 0.1)$ and two continuous variables $c_2, c_3 \sim U(-1, 1)$. During the optimization process and without the use of any labels the encoder learns to use c_1 to encode different digit classes, while c_2 and c_3 encode stroke width and digit



(a)

(b)

Figure 3.3: Images sampled from the (a) CelebA and (b) SVHN test sets. Each row shows images sampled according to one specific categorical variable c_{cat} which represents a learned characteristic.

rotation. Fig. 3.2a shows images randomly sampled from the test set according to the ten different categorical values. We can see that the encoder has learned to reliably assign a different categorical value for different digits. Indeed, by manually matching the different categories in c_1 to a digit type, we achieve a test set accuracy of 96.61% ($\pm$0.32%, averaged over 10 independent runs) without ever using labels during the training, compared to Chen et al. [2016] (unsupervised) with an accuracy of 95%, and Zhang et al. [2017a] (semi-supervised, 20 labels) with an accuracy of 96%. Fig. 3.2b shows images sampled from the test set for different values of c_2 and c_3. We see that we can use the encodings from E to now sample for digits with certain characteristics such as stroke width and rotation, even though this information was not explicitly provided during training.

On the CelebA data set the latent code is modeled with four categorical codes $c_1, c_2, c_3, c_4 \sim Cat(K = 10, p = 0.1)$ and four continuous variables $c_5, c_6, c_7, c_8 \sim U(-1, 1)$. Again, the encoder learns to associate certain image characteristics with specific codes in c. This includes characteristics such as the presence of glasses, hair color, and background color and is visualized in Fig. 3.3a.

On the SVHN data set we use the same network architecture and latent code

representations as for the CelebA data set. Again, the encoder learns interpretable, disentangled representations encoding characteristics such as image background, contrast and digit type. See Fig. 3.3b for examples sampled from the SVHN test set. These results indicate that the Bidirectional-InfoGAN is indeed capable of mapping data points into disentangled representations that encode meaningful characteristics in a completely unsupervised manner.

3.2.4 Conclusion

We showed that an encoder coupled with a generator in a Generative Adversarial Network can learn disentangled representations of the data without the need for any explicit labels. Using the encoder network we maximize the mutual information between certain parts of the generator's input and the images that are generated from it. Through this the generator learns to associate certain image characteristics with specific parts of its input. Additionally, the adversarial cost from the discriminator forces both the generator to generate realistic looking images and the encoder to approximate the inverse of the generator, leading to disentangled representations that can be used for inference.

The learned characteristics are often meaningful and humanly interpretable, and can potentially help with other tasks such as classification and clustering. Additionally, our method can be used as a pre-training step on unlabeled data sets, where it can lead to better representations for the final task. However, currently we have no influence over which characteristics are learned in the unsupervised setting which means that the model can also learn characteristics or features that are meaningless or not interpretable by humans. In the future, this can be mitigated by combining our approach with semi-supervised approaches, in which we can supply a limited amount of labels for the characteristics we are interested in to exert more control over which data-generating factors are learned while still being able to discover "new" generating factors which do not have to be known or specified beforehand.

3.3 Semi-supervised Learning of Disentangled Representations

This section presents our work *Image generation and translation with disentangled representations* by Tobias Hinz and Stefan Wermter published in 2018 at the *IEEE International Joint Conference on Neural Networks (pp. 5519 – 5526).*

3.3.1 Introduction

The introduction of Generative Adversarial Networks [Goodfellow et al., 2014] (GANs) provided a way to generate realistic images through a model that can be trained in an unsupervised fashion. While it has been observed that images

produced by GANs can be sharp and realistic, the original GAN model does not provide any control over what kind of image is generated. Furthermore, it does not provide a way to modify existing data samples, but can only generate new ones. Since then, GANs have been extended to also support or handle tasks such as image-to-image translation and controllable image generation, two tasks that require the modeling of high-dimensional data and a certain amount of understanding about the content of images.

Image-to-image translation takes as input some image and tries to "translate" it into a different domain. This can, for example, include changing the overall style of the image [Isola et al., 2017], translating the objects within the image into similar ones [Zhu et al., 2017a; Liu et al., 2017] or manipulating certain aspects of the image, e.g. by changing facial characteristics [Lample et al., 2017; Shen and Liu, 2017]. One difficulty in image-to-image translation is that it is often an unsupervised problem, i.e. we do not have a ground truth of what the translated image should look like. If we have, for example, the image of a male face and want to translate it into a female face, we usually have no image to compare it with and there are many different ways in which a male face could be modified to look more like a female one. Additionally, many current techniques need to train individual translators for each domain. This quickly becomes unfeasible as the number of domains increases, since for k domains $k(k-1)$ translators would be needed.

Controllable image generation is a related problem in which we want to exert some control over what kind of image is generated. This could for example mean specifying what kind of a digit is generated or whether a generated face should be male or female [Mirza and Osindero, 2014; Spurr et al., 2017; Zhang et al., 2017a]. This is usually achieved by providing a label to the generator and discriminator. Since the discriminator gets correctly labeled data samples from the real data distribution it learns to associate the labels with specific features in the images. In order to fool the discriminator the generator then learns to generate images that correspond to the provided labels. While this requires a (partially) labeled training set, it provides us with more control over what kind of images are generated and has also been shown to improve the image quality [Salimans et al., 2016].

So far, many of the methodologies focus on either image generation or image translation, but can rarely do both tasks. However, working in the domain of images there is conceptually not a big difference between translating images from one domain into another, or generating a new image according to certain conditions. Furthermore, many of the systems that perform image translation need distinct translators for individual domains – an approach that does not scale well with multiple domains. Additionally, many of the approaches need labeled training images for each of the domains they work with. Finally, most image translation methods encode the image information in entangled representations without easy access to the domain information or sometimes even exclude the domain information entirely from the image representation.

Our approach, on the other hand, aims at performing both, generating new images and translating between multiple domains with only one model. It does only need few labeled training samples and encodes all information into the representation

for the generator to use. Information important for the respective domains is encoded in a disentangled manner and we can even detect unknown data-generating factors without the need for any labels. For this, we make use of a generator G that generates an image X from a vector Z, and an encoder E that encodes images X into a representation Z. In order to gain control over the images, Z is divided into two parts (u, c). Here, u encodes image characteristics that we do not want to model explicitly, while c encodes characteristics that we want to control, e.g. which digit of the MNIST data set should be generated.

We then train the encoder to encode provided labels into c and all other information into u, while the generator is trained to construct realistic images from (u, c). This methodology already offers the possibility of translating images by using the encoder to get an image representation $Z = (u, c)$, changing c to the desired domain (e.g. changing the image class) and using the generator to generate the translated image. To also offer a way to generate new data samples we introduce a discriminator D that takes as input the representation Z and the according image X and tries to determine if the pair (Z, X) came from the generator or the encoder. Both the encoder and the generator try to build pairs (X, Z) that are classified incorrectly by the discriminator. This has two beneficial effects: firstly, it encourages the encoder and the generator to learn inverse functions of each other [Dumoulin et al., 2017] which should minimize the reconstruction error $X - G(E(X))$, and secondly, it forces the generator to generate meaningful and controllable images from randomly sampled Z.

To summarize: we propose a model that can do both controllable image generation and image-to-image translation between multiple domains based on information encoded within the representation. We only need very few labeled training examples (less than 2% on all tested data sets) and can even detect unknown data-generating factors, which can also be used for controllable image generation and translation. All information is directly encoded in the latent representation in a disentangled manner to which both the generator and the encoder have unrestricted access at all times during training.

3.3.2 Related Work

Mirza and Osindero [2014] introduced conditional GANs by supplying both the generator and the discriminator with labels. Perarnau et al. [2016] train two encoders with the help of a previously trained conditional generator where one encoder maps the image into a representation while the second encoder maps the image into a condition (e.g. the class label). Using the learned representation of an image a new condition can be specified to translate the original image. However, this approach only trains the encoder after the generator has already been trained, limiting the possibilities of interaction between the generator and the encoders. Additionally, all real data that is used during training the encoders needs to be labeled. Both Donahue et al. [2017] and Dumoulin et al. [2017] suggest training the encoder jointly at the same time as the generator. While this offers ways of encoding images, it does not offer any controllability over what kind of images are

generated, nor does it offer the possibility to translate images into other domains.

Chen et al. [2016] introduced GANs for disentangled representations. Training proceeds completely unsupervised and the model learns to disentangle underlying data-generating factors, which are modeled directly in the latent representation. These disentangled representations can then be used to control what kinds of images are generated. However, there is no control over which data-generating factors are learned and they do not necessarily coincide with human-interpretable factors (e.g. class labels). Spurr et al. [2017] extended this setting by incorporating some labels into the training process. Through this they have more control over which data-generating factors the generator learns, while still being able to also learn unlabeled or unknown data-generating factors. In addition to the generator and discriminator Li et al. [2017b] use a classifier to achieve a controllable generator. The data distribution is characterized by the classifier and the generator, while the discriminator only focuses on distinguishing between real and generated samples. However, all three systems lack the capability of mapping existing images into latent representations, i.e. they do not offer the possibility of image-to-image translation. Hinz and Wermter [2018b] introduced a model that learns disentangled representations for both generated and real data. While this model learns representations for existing data it can not translate between different domains and there is no control over which kind of data-generating factors are learned.

Zhang et al. [2017a] developed a model that is capable of both generating new images and translating existing ones. For this they split up the latent representations into two parts, encoding label information and other (unstructured) information. The labels are used for semi-supervised training and two additional inference networks are introduced which map images into the two parts of the representation. However, the model can only translate images based on the labels that are originally supplied, i.e. it cannot learn new data-generating factors and it needs individual inference networks for different parts of the representation. Choi et al. [2018] can also perform multi-domain image translation within one model, but require a fully labeled training set or the need to employ a specifically developed technique to deal with samples that are not fully labeled.

Shen and Liu [2017] model the image manipulation operation as learning the residual, i.e. learning the "difference image" between the original and the translated image. For this, they need two networks for each attribute in order to model inverse operations, e.g. one network to add eyeglasses and one network to remove glasses. Lample et al. [2017] use an encoder-decoder architecture for manipulating certain attributes of faces while keeping the rest of the image constant. This is achieved by making the representations invariant to the attributes, i.e. a representation encodes a face without the learned attributes and the generator gets the "base" representation plus the desired attributes. They need labels for each image during training and can also not detect novel data-generating factors. The procedure also imposes constraints on the representation, since it cannot encode the respective attributes which might have negative impacts on the quality of the learned representations.

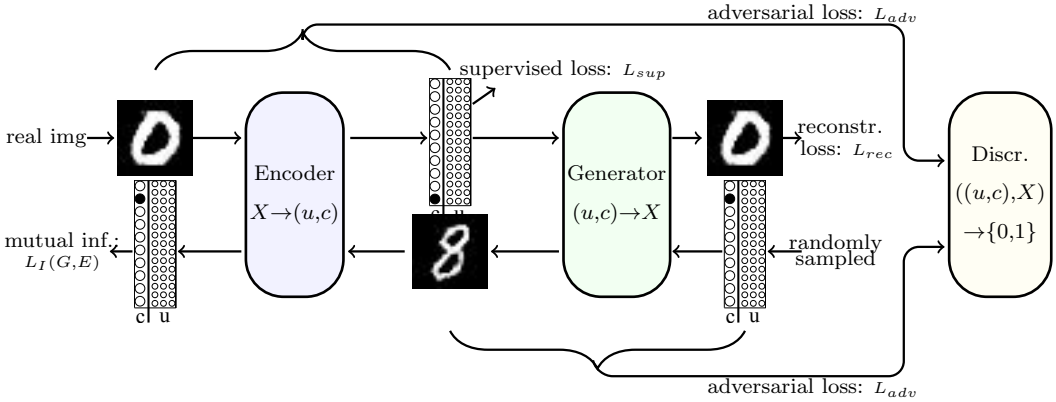


Figure 3.4: The encoder gets as input an image from the training set and possibly the associated label. It then encodes the image into a latent representation which is used by the generator to reconstruct the image. The generator gets as input a latent representation (either from the encoder or randomly sampled) and generates an image from it, while the encoder tries to reconstruct c from the image. The discriminator tries to determine whether pairs of latent representation and image come from the encoder or the generator.

3.3.3 Methodology

Our model consists of a generator G, an encoder E, and a discriminator D. The generator takes as input a vector Z and transforms it into an image X, while the encoder takes as input an image X and maps it into a latent representation Z. The discriminator takes as input both an image X and a latent representation Z, and tries to determine whether the pair came from the generator or the encoder. Figure 3.4 gives a high-level overview over our model. The latent representation Z is split up into two parts (u, c), where u encodes unstructured information and noise, while c encodes structured information and data-generating factors such as for example class labels.

G and E work together in that E takes as input an image, maps it into a latent representation, and G uses this representation to generate an image from it. Alternatively, the generator gets as input a randomly sampled representation Z and transforms it into an image from which the encoder tries to infer c. This is to ensure that the generator indeed uses the information provided in c as well as to detect previously unknown data-generating factors. The discriminator and its adversarial loss are used to improve the image quality and to encourage G and E to model inverse functions.

In order to ensure that the latent representation Z encodes the information that is needed to reconstruct the original image, we minimize the reconstruction loss, where E is the encoding part and G takes the role of the decoder:

$$\min_{G,E} L_{rec}(G,E) = \mathbb{E}_{X \sim P_{\text{data}}}[||X - G(E(X))||_2^2].$$

In contrast to other approaches, we do not condition the generator on additional

labels to control the image generation process, but instead encode all the necessary information directly within the latent representation Z, more specifically in c. To achieve this, c is made up of both categorical values c_{cat} and continuous values c_{cont} and the generator learns to associate these values with certain attributes or characteristics. To achieve this, we maximize the mutual information $I(c, G(u, c))$, i.e. the mutual information between c and the images generated from (u, c). Maximizing the mutual information directly is hard as it requires the posterior $P(c|x)$, and we therefore follow the approach by Chen et al. [2016] and define an auxiliary distribution $E(c|x)$ to approximate $P(c|x)$, where E is parameterized by our encoder. We then maximize the lower bound

$$\max_{G,E} L_I(G, E) = \mathbb{E}_{c\sim P(c),u\sim P(u),X\sim G(u,c)}[log\ E(c|X)] + H(c) \leq I(c; G(u, c)).$$

For simplicity reasons, we fix the distribution over c and, therefore, the entropy term $H(c)$ is treated as a constant.

While this approach is completely unsupervised and can detect meaningful characteristics that are encoded through c by itself, it has the drawback that we cannot specify certain characteristics that we want to be encoded within c (e.g. class labels). To remedy this, we introduce an additional cost, i.e. for a small subset of labeled data points we train E in a supervised manner to encode the provided information within c:

$$\min_{E} L_{sup}(E) = -\sum_{i} y_i\ log(y_i^*)$$

where y_i are labels such as class labels or characteristics like hair color or the presence of glasses and y_i^* are the encoder's predictions. Crucially, this process is only used to "guide" the encoder into associating certain specified information with given parts of c. Since the generator and the encoder are trained in a joint fashion (for maximizing the mutual information $I(c, G(u, c))$ and the reconstruction loss), the generator quickly picks up on the characteristics and their associated encodings and generates images with similar characteristics for similar encodings.

As a result, we only need comparatively few labeled data points for this process and the data points need not even be fully labeled. For example, if we have an image of a person's face with a given label that indicates the presence or absence of a smile, we can use this label to train the encoder to associate the presence or absence of a smile with a given part of c while ignoring the parts of c that encode other information. This also means that we can still use the model to learn to encode unlabeled characteristics within c while using other parts of c to encode predetermined characteristics.

Finally, to ensure that the images generated by G are realistic, that E and G learn to model inverse functions, and that E models u according to the chosen distribution we use a discriminator D. This discriminator tries to determine whether a pair of a latent representation and the associated image, i.e. either $((u, c), G(u, c))$ or $(X, E(X))$, comes from the encoder E or the generator G. Both

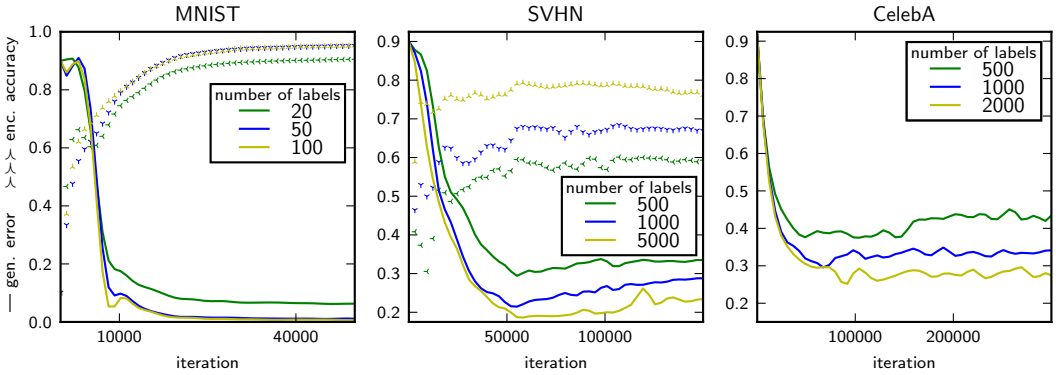


Figure 3.5: Development of generator and encoder accuracy averaged over ten independent runs for the different data sets. The dotted lines represent the encoder accuracy, while the continuous lines depict the error of the generator, i.e. percentage of inaccurately generated images.

E and G try to learn transformations that make the discriminator mis-classify their representation-image pair, i.e.

$$\min_{G,E} \max_{D} L_{adv}(D,G,E) = \mathbb{E}_{X\sim P_{\text{data}}}[logD(X,E(X))] + \mathbb{E}_{Z\sim P_Z}[log(1-D(G(Z),Z))].$$

This leads to our final objective function for training the whole model:

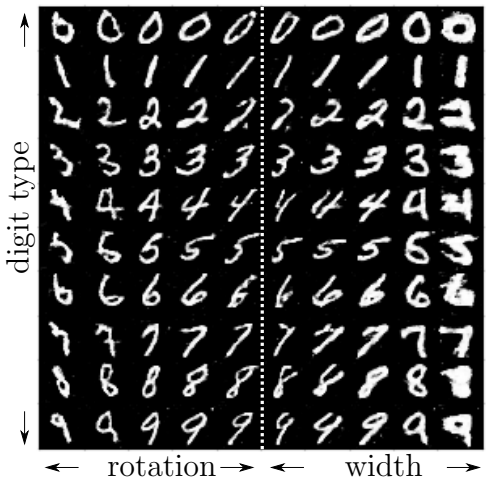
$$\min_{G,E} \max_{D} L(D,G,E) = \lambda_1 L_{sup} + \lambda_2 L_{rec} - \lambda_3 L_I + \lambda_4 L_{adv},$$

where $\lambda_i, i = 1...4$ are used to weight the impact of the individual loss terms. Given this model, we can now encode real or generated images into a latent representation $Z = (u, c)$. The characteristics learned in c are either specified through a limited amount of labeled training data or discovered in an unsupervised way by G and E. We can now use G to generate new images with given characteristics specified by c. We can also use G and E for image-to-image translation by obtaining the representation of a given image through E, modifying image characteristics by changing values in c, and generating the new image with updated image characteristics.

3.3.4 Experiments

We test our model on the MNIST, the SVHN, and the CelebA data set. On the MNIST and the SVHN data set the labeled information consists of class labels (digit type), while on the CelebA data set the labeled information contains attributes such as hair color and gender. For each data set we train a classifier to quantitatively assess the controllability of our generator and qualitatively examine the model's capability to translate images from the respective test sets.

On the MNIST data set we model u as a 16-dimensional vector, while c consists of a 10-dimensional categorical variable c_{cat} which encodes class information and



(a) In each column the random vector u is constant, while the categorical variable encoding class information changes for each row. The first five columns vary c_1 from -1 to 1, the second five columns vary c_2 from -1 to 1.

(b) The first column depicts randomly sampled images from the MNIST test set. The following columns are translated images where only the categorical value of c (encoding class information) was adapted.

Figure 3.6: (a) Image generation and (b) image translation, both performed by a model trained with 100 labels.

two continuous variables $c_1, c_2 \sim U(-1, 1)$. At regular intervals during the training process we generate 500 new images of each class by specifying c_{cat} accordingly and use a previously trained classifier (99.43% accuracy on the MNIST test set) to classify them. The development of the generator's accuracy (averaged over 10 independent runs) is depicted in Figure 3.5, and Table 3.1 gives the average accuracy at the end of training. We can see that for 50 and 100 samples the generator generates the desired digits with a very high accuracy and some generated samples are shown in Figure 3.6a.

Table 3.1: Errors (%) of generated samples on MNIST.

Model \ num labels	20	50	100
Triple-GAN [Li et al., 2017b]	3.06	1.80	1.29
SGAN [Zhang et al., 2017a]	1.68	1.23	0.93
Ours	6.29 ± 4.08	1.09 ± 0.89	0.66 ± 0.17

When we use only 20 labeled samples the generator sometimes "mixes up" two classes (e.g. 4 and 9), which then leads to a lower accuracy of only roughly 80%. The chances of this happening are around 50% and this explains the comparatively low generator accuracy when only 20 labels are used. This problem does not occur when marginally more samples are used and for 50 labeled samples this problem did not occur anymore. Zhang et al. [2017a] achieve a good performance even with 20 labels but have to use two distinct inference networks, one for the class information and one for the unstructured part of the representation. Spurr et al. [2017], who

only focus on learning a controllable generator, report that they need a minimum of 132 labels to achieve their goal.

Not only do we have control over what kind of digit is generated by the generator, but we can also control the stroke width and digit rotation by modifying c_1 and c_2. These characteristics were identified without any labels by maximizing the mutual information between c_1, c_2, and the images generated from them. Figure 3.6a also shows these characteristics by interpolating c_1 from -1 to 1 in the first five columns and c_2 in the last five columns.

Figure 3.5 also shows the development of the encoder's accuracy measured on the test set. We can see that the accuracy approaches around 96% for 50 and 100 labels. While this is not the optimal known state-of-the art for this number of labels we still achieve reasonable performance even though this is not our main training criterion. Other approaches that outperform ours with the same amount of labels usually focus explicitly on the encoder accuracy and the system is trained to determine the digit classes. Our model, on the other hand, does not only encode the digit identity, but also other information about the image such as general style, stroke width, and digit rotation.

Figure 3.7: Image interpolation: the first and last columns are images sampled from the MNIST test set, the intermediate columns are linear interpolations.

Finally, we show that we can translate images from the test set into other classes in Figure 3.6b and interpolations between test set images in Figure 3.7. The first column in Figure 3.6b depicts images from the MNIST test set, while the other columns show the translation of that image to other digit classes. We can see that general style information as well as stroke width and digit rotation are consistent across the translations. In Figure 3.7, the first and last columns show images from the MNIST test set, while the intermediate columns depict linear interpolations between the two images. We see that the interpolations progress smoothly and often change class identities around midway through the interpolation, when the "class label" in the representation has the same probability for the respective start and end classes.

On the SVHN data set we model u as a 128-dimensional vector, while c consists of four categorical variables and four continuous variables. We use c_{cat_1} to encode the class information (10 classes), while the other three categorical variables are five-dimensional. Again, we track the performance of the generator during training with the help of a previously trained classifier (94.51% accuracy on the test set). Figure 3.5 shows that the accuracy of both the generator and the encoder increases with the amount of labels,

Table 3.2: Errors (%) of generated samples.

Data Set	SVHN			CelebA		
Number of Labels	500	1000	5000	500	1000	2000
Error (%)	32.64	25.61	18.76	37.43	30.92	26.95
Standard Deviation	4.36	2.06	3.32	3.10	5.30	2.75

(a) Representations are kept constant in each row while the variable encoding class information changes across the columns.

(b) The first and last columns are images sampled from the SVHN test set, the intermediate columns are linear interpolations.

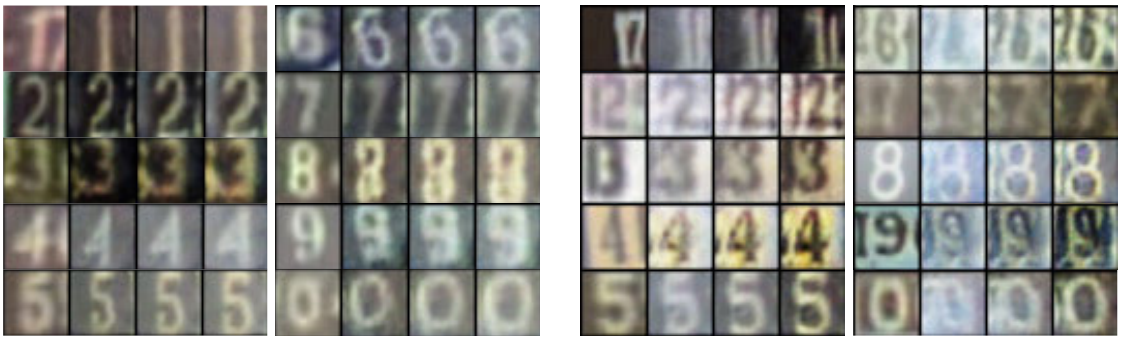
Figure 3.9: (a) Image generation and (b) image interpolation, both performed by a model trained with 1000 labels.

while Table 3.2 shows the average performance of the generator after training is completed. We achieve a reasonably good performance with only 1000 labeled samples, as opposed to Spurr et al. [2017] who need more than 7000 labeled samples to achieve a similar generator accuracy.

Figure 3.9a shows samples of different classes generated by the generator. We can see that stylistic information is kept constant across the different classes. This indicates that c_{cat_1} indeed captures the label information, while the rest of the representation Z encodes other stylistic information. Figure 3.8 shows translations of images from the SVHN test set. Again, we can see that the stylistic information is conserved across the different images, while the digit is controlled by c_{cat_1}. Figure 3.10 shows translations where only one of the continuous variables is changed across columns. Here, we show examples of two of the continuous variables, which learned to encode the digit size and the contrast. These factors were learned completely unsupervised and without any supplied labels. Finally, Figure 3.9b shows interpolations between images from the test set. Again, the interpolations progress in a smooth manner and, as before, the digit identity changes around midway through the interpolation.

Figure 3.8: Image translation: the first column contains randomly sampled images from the SVHN test set, while the other columns are translated images where the variable encoding class information is changed across columns.

On the CelebA data set we also model u as a 128-dimensional vector. Since the CelebA data set has no class labels as such, we choose to encode the facial



digit size contrast

Figure 3.10: Image translation: the first column of each block shows randomly sampled images from the SVHN test set. The other columns are translated images where one of the continuous variables c_{cont} is changed from -1 to 1.

attributes hair color, gender, smiling and pale skin within c. It therefore consists of a five-dimensional categorical variable c_{cat_1} encoding hair color (bald, black, blond, brown, gray), and three two-dimensional categorical variables for the other three attributes. Additionally, we use four continuous variables to encode other (unknown) characteristics. We make sure that our labeled subset of training data is roughly balanced according to the individual labels. This means that approximately one fifth of the labeled samples are sampled from each of the five hair colors, while also ensuring that the samples are split somewhat evenly for the other attributes. The one exception to this rule is the class of *bald* faces, since they are all male.

Figure 3.5 shows how the accuracy of the generator develops for different amounts of labels averaged over 10 independent runs and detailed results can be found in Table 3.2. The graph only shows the accuracy for generated images according to the characteristics *black, blond, brown hair; smiling, not smiling; male, female*. This is because we found that the generator often fails to generate the characteristics *bald* and *gray hair* correctly. Additionally, images labeled with *pale skin* in the CelebA data set tend to be "very" pale. While our generator generates images with pale skin, it was usually not "pale enough" for our classifier to classify correctly and it was therefore also left out, but can be qualitatively seen in the generated images. Due to this, we also do not depict the accuracy of the encoder on the training data, since the labels on the CelebA data set are not always accurate and sometimes contradict each other (e.g. there are also images that are labeled with multiple hair colors). Still, for the mentioned characteristics we achieve a good accuracy with only 2000 samples, whereas Spurr et al. [2017] report the need for more than 30000 images to achieve a stable training performance.

Figure 3.11 shows images generated according to different settings of c, while Figure 3.13 shows image translations according to the same characteristics. We can see that the characteristic *bald* does not work very well (especially for women). The reason for this is most likely that the labeling on the CelebA data set is not always correct and many images that are labeled as *bald* are not actually bald (more than 25% of the images labeled as *bald* are also labeled as *gray hair* and there are no bald women in the CelebA data set). We also observe that the attribute *gray hair*

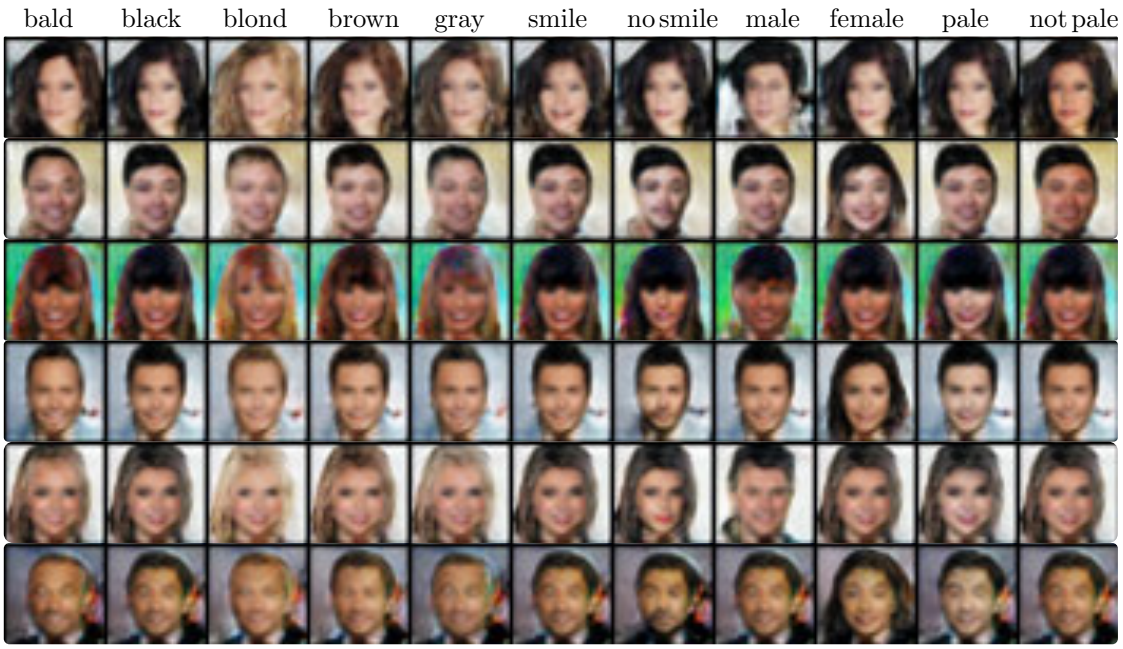
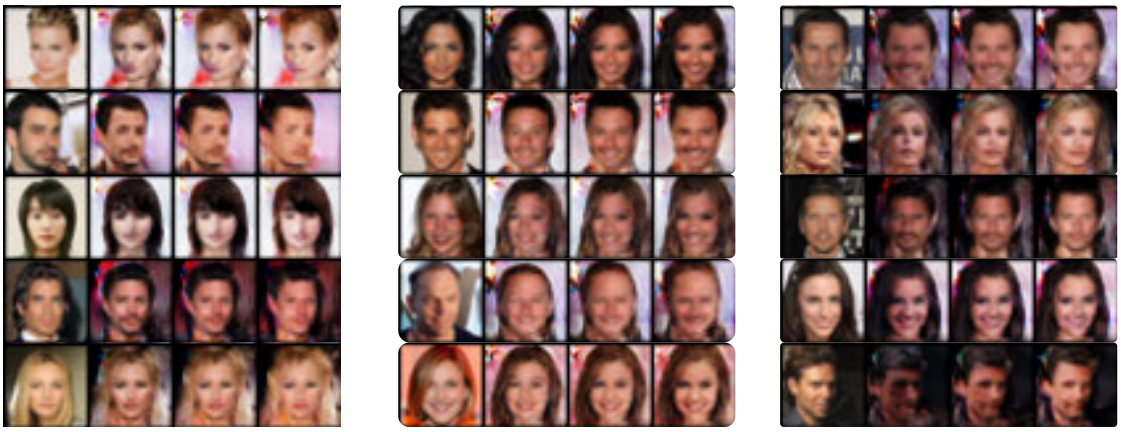


Figure 3.11: Image generation (model trained with 1000 labels): in each row the latent representation u is kept constant and only the categorical variables are changed across the columns.

often leads the generator to increase the depicted person's age, since there is a high correlation between *gray hair* and age in the CelebA data set (only 380 of 8499 images with the label *gray hair* are also labeled as *young*).

On the other hand, the characteristic *blond hair* works better for women than for men. Again, this is most likely due to the fact that more women than men in the CelebA data set are blond (28234 vs. 1749) and blond hair tends to be more "pronounced" for women. Other characteristics such as *smiling* and gender are modeled very accurately. Finally, Figure 3.12 shows translations according to some of the continuous variables which were trained completely without labels. We can see that they learned to encode information such as the amount of applied make-up, the size of the face, and skin tone.

Finally, the image translations do not always translate the face identity correctly, i.e. we can see that the facial features change slightly between the original image and the translated ones. This is most likely due to the fact that our reconstruction loss is only a small part of the overall loss and the adversarial loss is usually the dominating factor. As a result, high level features such as e.g. facial orientation are reconstructed correctly, while more detailed features are sometimes lost. We find that we can increase the fidelity of the translations by increasing the weight λ_2 of the reconstruction loss, however, this leads to a slight drop in the accuracy of the generator. At the moment there therefore exists a trade-off between the fidelity of the translations and the accuracy of the generator.



make-up face size skin tone

Figure 3.12: Image translation: the first column of each block contains randomly sampled images from the CelebA test set. The other columns are translated images where only one of the continuous variables is changed from -1 to 1.

original bald black blond brown gray smile no smile male female pale not pale

Figure 3.13: Image translation: the first column contains randomly sampled images from the CelebA test set while the other columns are translated images where individual categorical variables are changed across columns.

3.3.5 Conclusion

We developed a system that is capable of both image-to-image translation and controllable image generation. We make use of an encoder which encodes existing images into a latent representation and a generator which takes as input a latent representation and generates an image from it. The latent representation is split up into two parts encoding unstructured information and structured information such as class labels. The structured information is encoded in a disentangled manner and by maximizing the mutual information between this disentangled representation

and the images generated from it, we can detect unknown data-generating factors. By specifying the structured information, the image generation process can be controlled, making it useful for generating new images and translating images by adapting the latent representation obtained through the encoder. A discriminator taking as input both a representation and the corresponding image improves the quality of the generated images and encourages the encoder and the decoder to learn inverse function of each other.

Compared to other state-of-the-art image generation and translation systems we can do both, controllable image generation and image-to-image translations with one model. We can translate between multiple domains without the need for multiple encoder-decoder pairs and can even detect novel data-generating factors, which can then be used as additional information for the image generation and translations tasks. For all this we only need a small amount of labeled training samples and we can also make use of only partially labeled data. We test the system on the MNIST, SVHN, and CelebA data sets and show that it is capable of both image translation and controllable image generation across these different data sets. We also find that the system does learn unlabeled data-generating factors such as digit size and rotation or image contrast, which enables us to generate and translate images where we can specify these characteristics, even without any labels provided for them in the training data.

3.4 Intermediate Discussion

Section 3.2 and Section 3.3 show how we can use GANs to learn and use disentangled representations. While we show good results in both unsupervised and semi-supervised settings, learning and using disentangled representations comes with several challenges which we will discuss here. Some of these challenges come from the difficulty of defining and evaluating disentangled representations (see Section 3.1). Additionally, we look into the question of whether disentangled representations can be learned in a completely unsupervised fashion in the first place Locatello et al. [2019b] and how useful disentangled representations actually are for downstream tasks [Locatello et al., 2019a; Van Steenkiste et al., 2019].

Unsupervised Learning of Disentangled Representations In Section 3.2 we show how we learn disentangled representations in an unsupervised fashion and Section 3.3 shows additional results of attributes that were learned without any labels. However, recent work shows that it may be impossible to learn disentangled representations in a completely unsupervised manner and that, instead, inductive biases are needed for both the model and the data set [Locatello et al., 2019b]. These inductive biases take the form of, e.g., choosing data sets that may have clearly interpretable factors and tuning hyperparameters of the chosen models until they learn the given factors.

This is indeed the case for both our approaches (Section 3.2 and Section 3.3). We train our models on commonly used data sets (MNIST [LeCun et al., 1998], SVHN [Netzer et al., 2011], and CelebA [Liu et al., 2015]) which contain "simple"

objects (digits or faces) that have humanly interpretable attributes (e.g. class labels for digits and hair color for faces). We choose our hyperparameters for the different models (model architecture, optimization parameters, size and form of the disentangled representation, ...) manually such that we achieve "good" disentangled representations. For this, we train the models with different hyperparameters and visually inspect the results until we find hyperparameters that result in interpretable representations that learn characteristics that we expect (e.g. class labels). Through this, we introduce a strong implicit bias in the sense that we define what we want the model to learn on a meta-level and then fine-tune the model's hyperparameters in an external loop until we reach the desired results.

If we only look at the loss that is optimized during training, and specifically the mutual information which is our proxy for learning disentangled representations, we observe that the models reach very high mutual information between images and learned representations for many hyperparameter settings. Therefore, based only on the mutual information, it would seem that the model has learned disentangled representations. However, if we visually inspect the learned representations and look for the results that we desire a priori, only a small subset of the hyperparameters actually learn disentangled representations as expected by us.

This shows the challenges of working with disentangled representations while lacking a clear definition of them. Instead, for many data sets the community has converged on a set of characteristics that a disentangled representation should model, and then proceeded to fine-tune the hyperparameters of different models until the desired goal is reached. Indeed, Locatello et al. [2019b] show that disentangled representations can not be found without access to ground truth labels, that "good" model hyperparameters often do not transfer to other data sets, and that even seemingly innocuous things such as the random seed chosen at train time might influence the form of the learned disentangled representation.

On the other hand, disentangled representations occur, to a degree, consistently in deep neural networks through regularization and other implicit biases such as enforcing translation and scale invariance [Achille and Soatto, 2018]. The hypothesis is that this is somewhat related to the information bottleneck [Tishby et al., 1999; Tishby and Zaslavsky, 2015] in the sense that learning disentangled representations can help with compression. However, whether the resulting representations are indeed "disentangled" is again strongly dependent on the used definition. As such, whether disentangled representations can actually be learned in a completely unsupervised manner might ultimately dependent on the definition of "disentanglement" in the given context.

Evaluating Disentangled Representations As discussed previously in Section 3.1, evaluating disentangled representations is also a challenging task. In Section 3.2 we evaluate the disentangled representations quantitatively on the MNIST data set by using the disentangled representation to predict the digit within each image. All other results were evaluated visually. We performed more quantitative analyses in Section 3.3 by using pre-trained convolutional neural networks to evaluate generated images. However, as alluded to in the previous paragraph, all of

our evaluation was done either visually or based on pre-defined, manually chosen characteristics that we wanted to be represented in a disentangled manner. As a consequence, our quantitative evaluation is limited to factors of variation which we were aware of before training the networks. We also show that it is possible to discover previously unknown factors of variation through visual inspection (see e.g. Figure 3.12). However, no quantitative evaluation for these characteristics was performed in our experiments and different persons might judge and evaluate the unsupervised learned characteristics differently.

Locatello et al. [2019b, 2020a] show that it is challenging to choose which model performs better than others for disentangled representation learning on a given data set. Different evaluation metrics give different results, partly due to the unclear definition of what disentangled representations are in the first place. Visual inspection of the learned representations is biased, hard to reproduce, and relies on the made assumptions about the data generating factors. As such, arriving at a generally accepted way to evaluate disentangled representations might not be possible until a generally applicable definition for disentangled representations is available. Until then, most evaluations will likely rely on visual inspections or use metrics and evaluation procedures specifically for the intended downstream task.

Disentangled Representations for Downstream Tasks One of the central hypotheses with respect to disentangled representations is that they are useful for many different downstream tasks [Bengio et al., 2013]. While several methods have shown that the learned representations can be useful for downstream tasks such as controllable image synthesis, most of these methods are evaluated on tasks that were known in advance. As discussed previously, this means that these approaches were most likely fine-tuned on a meta-level by the respective researchers to learn representations that perform well on the chosen task.

But what if the task is not known a priori? Are disentangled representations still useful for a range of tasks that they were not specifically trained for? Locatello et al. [2019b, 2020b] do not find that disentangled representations are always helpful for downstream tasks despite achieving good scores on several evaluation metrics for disentangled representations. Van Steenkiste et al. [2019] find that disentangled representations improve the abstract visual reasoning capability of several models. It seems likely that disentangled representations will be helpful for some tasks, but not all tasks. However, which tasks generally benefit from disentangled representations (without knowing the task at train time) is difficult to predict and will likely depend on the used definition of disentanglement, the implicit biases that are present in the model design and data set, and the used evaluation metrics.

3.5 Summary

We showed how we can learn disentangled representations in an unsupervised (Section 3.2) or semi-supervised manner (Section 3.3) and what we can do with these disentangled representations on several data sets. However, in Section 3.4

we identified several challenges in the definition and evaluation of disentangled representations that also apply to our approaches. The "unsupervised" learning of disentangled representations is not actually unsupervised but depends on the choice of model architecture, hyperparameters, and data set, all of which is done manually. As such, the unsupervised learning is actually supervised on a meta-level and the learning is only unsupervised in the inner loop of model optimization. Furthermore, the evaluation of disentangled representations has several challenges and, to date, there is no clear "winner" in the field of metrics for evaluating disentangled representations. Instead, evaluation is often done manually based on visual results and depends on the practitioner's definition of disentanglement.

However, even if disentanglement is only defined individually for different tasks and data sets, it is still desirable for specific tasks and interpretability. Even if the definition of disentanglement might not transfer between different domains and different evaluation metrics might be needed for different tasks and models, disentanglement can still be useful when used correctly. Several approaches have shown that in specific domains we can achieve representations that encode interpretable factors in distinct parts of the representations. If we have a concrete goal (e.g. image editing) with concrete priors that we want to be encoded in a disentangled manner, then this is certainly possible (to a degree). While we may not be able to claim that learning disentangled representations is a completely unsupervised manner is possible, future work should focus on how we can learn disentangled representation with as little prior knowledge and implicit biases as possible, how we can unite different notions of disentanglement across tasks and data sets, and how we can go beyond visual evaluation of disentanglement.

Another challenge is that it is often unclear which degree of disentanglement is desired. The original definition we used implies that disentangled representations should model all data generating factors, however small or negligible they might be. This would not only mean disentangling foreground from background, or one object from another object but also that all factors that model each individual object (e.g. size, shape, texture) should be disentangled. Once we introduce more than one object into the environment this becomes increasingly complex, especially since objects differ in their complexity and, thus, the number of data-generating factors per object. One approach towards dealing with complex environments would be to model the environment in a compositional manner where each object is represented explicitly via its own representation. We follow this approach in Chapter 4 and introduce several approaches for learning compositional representations that are capable of explicitly modeling individual objects. In the future, these explicit object representations could be combined with the approaches developed and discussed in this chapter in order to learn disentangled object representations that can be combined to model complex and realistic environments.

Chapter 4

Compositional Representations

As we discussed in Chapter 3, learning disentangled representations becomes increasingly difficult once multiple objects and complex environments are introduced. To address this, it might be useful to model objects individually instead of learning a single representation for some global input. In this chapter, we introduce our approaches to learning compositional representations which we define as representations that explicitly model individual parts within a larger context. These distinct representations can then be combined to create more complex contexts. We illustrate compositional representations in the setting of complex visual scenes that contain several individual objects. This is in contrast to Chapter 3 where the context always consists of a single object from one domain (e.g. digits or faces). We first describe what compositional representations are in our setting and why they might be useful. After this, we show our approach to learning compositional representations and how they can be evaluated. We conclude with a critical discussion and place our approaches within the context of current literature.

4.1 Compositional Representations

Many image data sets are neatly pre-processed such that they only contain images that show a single object located in the image center. This is often referred to as "center bias" [Tseng et al., 2009]. However, the real world is more complex than that, and images often contain different objects which might partially occlude or interact with each other and are often not positioned in the image center. Rather than disentangling all these individual objects and interactions within a single representation it makes sense to learn individual representations for each of the objects. These individual representations can then be composed to represent complex visual scenes.

This is similar to humans, who naturally decompose complex scenes into individual objects and model boundaries between objects and possible interactions [Spelke, 1990; Hupé et al., 1998]. Humans not only decompose scenes into individual objects but also have an additional understanding of how individual objects are made up of smaller parts [Hoffman and Richards, 1984; Biederman, 1987]. Indeed, Lake

et al. [2017] claim that compositionality is one of the central paradigms of human intelligence and Greff et al. [2020] identify compositionality as a key ingredient for generalization. Early computer vision models have been inherently compositional [Fidler and Leonardis, 2007], however, most current computer vision models rely on deep neural networks which do not model concepts in a compositional manner [Tokmakov et al., 2019; Greff et al., 2019]. There are, to date, very few neural network models that address compositionality and evaluation of compositionality for computer vision and deep neural networks is still in its infancy [Andreas, 2019].

Disentangled representations (Chapter 3) learn one representation for one input where different characteristics of the input are represented in disjoint parts of the representation. In contrast to this, compositional representations model individual parts of a context explicitly in different representations. As such, we can use compositional representations to model e.g. images that either contain several objects of a single domain (e.g. several digits distributed across an image) or, even more challenging, images that contain several objects of distinct domains (e.g. humans, cars, and dogs). Other approaches have already made use of the advantages of compositional representations, though, similar to disentangled representations, there is no clear definition of them. We now give a brief overview of different approaches to compositional representations before describing how we define and use them in our work.

Part-based Compositional Representations Part-based compositional representations [Yuille and Mottaghi, 2016] model individual objects via their characteristics or parts, e.g. by representing a given object through a composition of color and shape. Mobahi et al. [2014] use compositional representations for tasks such as image morphing and scene alignment. Their compositional representations model color, appearance, and shape of a given image, and the three compositional parts can be reused between tasks and images. Tabernik et al. [2016] define representations as compositional if representations at a later modeling stage explicitly model combinations of representations at early stages, while Tang et al. [2017] model compositionality through an And-Or Graph that characterizes subpart-part compositions. Wang and Yuille [2015] use semantic segmentation maps to learn part-based compositional representations of animals while Tokmakov et al. [2019] use category-level attribute annotations in order to decompose representations of objects into visual parts such as color or more abstract concepts such as symmetry. However, relying only on part-based representations does not automatically lend itself to the extended use-case of modeling several objects within a given context.

Object-based Compositional Representations Object-based compositional representations model complex scenes by representing individual objects explicitly. These individual object representations can then be used to compose novel scenes. Chang et al. [2017] use compositional representations to model individual objects in complex scenes to predict future object states. Each representation for a given object encodes various object properties such as position, velocity, and mass. With the help of these representations, they can model physical interactions between individual objects based on current and past object states. Stone et al. [2017]

model compositionality by training representations to be partially invariant to individual objects by introducing a novel loss function. What constitutes an object is defined through labeled masks at train time. Gao et al. [2016] use videos to learn object-based representations. In this case, the learning signal comes from temporally nearby image frames which are used to train the model to learn similar representations for object-like regions. Several current approaches try to learn unsupervised object-based representations from multiple images based on single [Greff et al., 2019; Burgess et al., 2019; Engelcke et al., 2020] or multiple views [Li et al., 2020a] of the same scene. While these approaches explicitly model several objects within a given context, they are often based on discriminative frameworks. In contrast, our approaches use generative models for learning to model images that contain multiple objects.

Compositional Representations in Our Approaches Our approaches use object-based compositional representations to model complex scenes that contain several objects. We represent these individual objects explicitly by using different model parameters to model either objects or the image background. In this context, a given model learns representations where each representation R consists of a set of sub-representations r_i for different objects i and one representation r_b for the image background: $R = \cup_i r_i \cup r_b$. These representations R are what we refer to as compositional representations.

We partly address the second view of compositional representations (part-based) by conditioning each object representation r_i not only on an object class but also on additional attributes (where available) such as color and position. Note that our approaches to compositional representations are supervised and need labeled data for what kind of objects are present in a given scene. Nevertheless, our approaches are some of the first generative models that explicitly represent several objects in a given scene with compositional representations.

Advantages of Compositional Representations Compositional representations have several advantages compared to traditional representations. As we do not learn one single representation that encodes knowledge about the whole context the learning itself becomes easier, as individual representations focus on individual objects. Compositional representations also lend themselves to representing contexts in a compositional way. They also offer the advantage of reusability, as representations of a given object – once learned – can be used in different contexts. Finally, compositional representations might be useful for tasks such as visual reasoning, as interactions between individual objects can be represented explicitly via relations between different representations [Locatello et al., 2020c].

4.2 Learning Compositional Representations

This section presents our work *Generating multiple objects at spatially distinct locations* by Tobias Hinz, Stefan Heinrich, and Stefan Wermter published in 2019 at the *International Conference on Learning Representations*.

4.2.1 Introduction

Understanding how to learn powerful representations from complex distributions is the intriguing goal behind adversarial training on image data. While recent advances have enabled us to generate high-resolution images with Generative Adversarial Networks (GANs), currently most GAN models still focus on modeling images that either contain only one centralized object (e.g. faces (CelebA), objects (ImageNet), birds (CUB-200), flowers (Oxford-102), etc.) or on images from one specific domain (e.g. LSUN bedrooms, LSUN churches, etc.). This means that, overall, the variance between images used for training GANs tends to be low [Raj et al., 2017]. However, many real-life images contain multiple distinct objects at different locations within the image and with different relations to each other. This is for example visible in the MS-COCO data set [Lin et al., 2014], which consists of images of different objects at different locations within one image. In order to model images with these complex relationships, we need models that can model images containing multiple objects at distinct locations. To achieve this, we need control over what kind of objects are generated (e.g. persons, animals, objects, etc.), the location, and the size of these objects. This is a much more challenging task than generating a single object in the center of an image.

Current work [Karacan et al., 2016; Johnson et al., 2018; Hong et al., 2018b; Wang et al., 2018a] often approaches this challenge by using a semantic layout as additional conditional input. While this can be successful in controlling the image layout and object placement, it also places a high burden on the generating process since a complete scene layout must be obtained first. We propose a model that does not require a full semantic layout, but instead only requires the desired object locations and identities (see Figure 4.1). One part of our model, called the global pathway, is responsible for generating the general layout of the complete image, while a second path, the object pathway, is used to explicitly generate the features of different objects based on the relevant object label and location.

The generator gets as input a natural language description of the scene (if existent), the locations and labels of the various objects within the scene, and a random noise vector. The global pathway uses this to create a scene layout encoding which describes high-level features and generates a global feature representation from this. The object pathway generates a feature representation of a given object at a location described by the respective bounding box and is applied iteratively over the scene at the locations specified by the individual bounding boxes. We then concatenate the feature representations of the global and the object pathway and use this to generate the final image.

The discriminator, which also consists of a global and object pathway, gets as input the image, the bounding boxes and their respective object labels, and the textual description. The global pathway is then applied to the whole image and obtains a feature representation of the global image features. In parallel, the object pathway focuses only on the areas described by the bounding boxes and the respective object labels and obtains feature representations of these specific locations. Again, the outputs of both the global and the object pathway are merged

and the discriminator is trained to distinguish between real and generated images.

In contrast to previous work we do not generate a scene layout of the whole scene but only focus on relevant objects which are placed at the specified locations, while the global consistency of the image is the responsibility of the other part of our model. To summarize our model and contributions: 1) We propose a GAN model that enables us to control the layout of a scene without the use of a scene layout. 2) Through the use of an object pathway which is responsible for learning features of different object categories, we gain control over the identity and location of arbitrarily many objects within a scene. 3) The discriminator judges not only if the image is realistic and aligned to the natural language description, but also whether the specified objects are at the given locations and of the correct object category. 4) We show that the object pathway does indeed learn relevant features for the different objects, while the global pathway focuses on general image features and the background.

4.2.2 Related Work

Having more control over the general image layout can lead to a higher quality of images [Reed et al., 2016a; Hong et al., 2018b] and is also an important requirement for semantic image manipulation [Hong et al., 2018a; Wang et al., 2018a]. Approaches that try to exert some control over the image layout utilize Generative Adversarial Nets [Goodfellow et al., 2014], Refinement Networks (e.g. Chen and Koltun [2017]; Xu et al. [2018a]), recurrent attention-based models (e.g. Mansimov et al. [2016]), autoregressive models (e.g. Reed et al. [2016c]), and even memory networks supplying the image generation process with previously extracted image features [Zhang et al., 2018c].

One way to exert control over the image layout is by using natural language descriptions of the image, e.g. image captions, as shown by Reed et al. [2016b], Zhang et al. [2018a], Sharma et al. [2018], and Xu et al. [2018b]. However, these approaches are trained only with images and their respective captions and it is not possible to specifically control the layout or placement of specific objects within the image. Several approaches suggested using a semantic layout of the image, generated from the image caption, to gain more fine-grained control over the final image. Karacan et al. [2016], Johnson et al. [2018], and Wang et al. [2018a] use a scene layout to generate images in which given objects are drawn within their specified segments based on the generated scene layout. Hong et al. [2018b] use the image caption to generate bounding boxes of specific objects within the image and predict the object's shape within each bounding box. This is further extended by Hong et al. [2018a] by making it possible to manipulate images on a semantic level. While these approaches offer a more detailed control over the image layout they heavily rely on a semantic scene layout for the image generating process, often implying complex preprocessing steps in which the scene layout is constructed.

The two approaches most closely related to ours are by Reed et al. [2016a] and Raj et al. [2017]. Raj et al. [2017] introduce a model that consists of individual "blocks" which are responsible for different object characteristics (e.g. color, shape,

etc.). However, their approach was only tested on the synthetic SHAPES data set [Andreas et al., 2016], which has only comparatively low variability and no image captions. Reed et al. [2016b] condition both the generator and the discriminator on either a bounding box containing the object or keypoints describing the object's shape. However, the used images are still of relatively low variability (e.g. birds [Wah et al., 2011]) and only contain one object, usually located in the center of the image. In contrast, we model images with several different objects at various locations and apply our object pathway multiple times at each image, both in the generator and in the discriminator. Additionally, we use the image caption and bounding box label to obtain individual labels for each bounding box, while Reed et al. [2016b] only use the image caption as conditional information.

4.2.3 Approach

For our approach, the central goal is to generate objects at arbitrary locations within a scene while keeping the scene overall consistent. For this we make use of a generative adversarial network (GAN) [Goodfellow et al., 2014]. A GAN consists of two networks, a generator and a discriminator, where the generator tries to reproduce the true data distribution and the discriminator tries to distinguish between generated data points and data points sampled from the true distribution. We use the conditional GAN framework, in which both the generator and the discriminator get additional information, such as labels, as input. The generator G (see Figure 4.1) gets as input a randomly sampled noise vector z, the location and size of the individual bounding boxes $bbox_i$, a label for each of the bounding boxes encoded as a one-hot vector l_{onehot_i}, and, if existent, an image caption embedding φ obtained with a pretrained char-CNN-RNN network from Reed et al. [2016b]. As a pre-processing step (A), the generator constructs labels $label_i$ for the individual bounding boxes from the image caption φ and the provided labels l_{onehot_i} of each bounding box. For this, we concatenate the image caption embedding φ and the one-hot vector of a given bounding box l_{onehot_i} and create a new label embedding $label_i$ by applying a matrix-multiplication followed by a non-linearity (i.e. a fully connected layer). The resulting label $label_i$ contains the previous label as well as additional information from the image caption, such as color or shape, and is potentially more meaningful. In case of missing image captions, we use the one-hot embedding l_{onehot_i} only.

The generator consists of two different streams which get combined later in the process. First, the *global pathway* (B) is responsible for creating a general layout of the global scene. It processes the previously generated local labels $label_i$ for each of the bounding boxes and replicates them spatially at the location of each bounding box. In areas where the bounding boxes overlap the label embeddings $label_i$ are summed up, while the areas with no bounding boxes remain filled with zeros. Convolutional layers are applied to this layout to obtain a high-level layout encoding which is concatenated with the noise vector z and the image caption embedding φ and the result is used to generate a general image layout f_{global}.

Second, the *object pathway* (C) is responsible for generating features of the

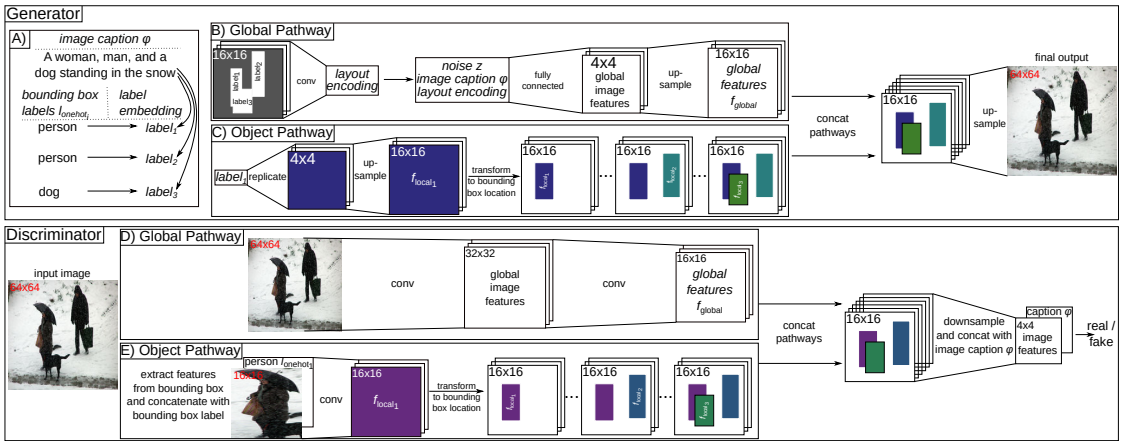


Figure 4.1: Both the generator and the discriminator of our model consist of a global and an object pathway. The global pathway focuses on global image characteristics, such as the background, while the object pathway is responsible for modeling individual objects at their specified location.

objects f_{local_i} within the given bounding boxes. This pathway creates a feature map of a predefined resolution using convolutional layers which receive the previously generated label $label_i$ as input. This feature map is further transformed with a Spatial Transformer Network (STN) [Jaderberg et al., 2015] to fit into the bounding box at the given location on an empty canvas. The same convolutional layers are applied to each of the provided labels, i.e. we have one object pathway that is applied several times across different labels $label_i$ and whose output feeds onto the corresponding coordinates on the empty canvas. Again, features within overlapping bounding box areas are summed up, while areas outside of any bounding box remain zero.

As a final step, the outputs of the global and object pathways f_{global} and f_{local_i} are concatenated along the channel axis and are used to generate the image in the final resolution, using common GAN procedures. The specific changes of the generator compared to standard architectures are the object pathway that generates additional features at specific locations based on provided labels, as well as the layout encoding which is used as additional input to the global pathway. These two extensions can be added to the generator in any existing architecture with limited extra effort.

The discriminator receives as input an image (either original or generated), the location and size of the bounding boxes $bbox_i$, the labels for the bounding boxes as one-hot vectors l_{onehot_i}, and, if existent, the image caption embedding φ. Similarly to the generator, the discriminator also possesses both a *global* (D) and an *object* (E) pathway respectively. The global pathway takes the image and applies multiple convolutional layers to obtain a representation f_{global} of the whole image. The object pathway first uses a STN to extract the objects from within the given bounding boxes and then concatenates these extracted features with the spatially replicated bounding box label l_{onehot_i}. Next, convolutional layers are applied and

the resulting features f_{local_i} are again added onto an empty canvas within the coordinates specified by the bounding box. Note, similarly to the generator we only use one object pathway that is applied to multiple image locations, where the outputs are then added onto the empty canvas, summing up overlapping parts and keeping areas outside of the bounding boxes set to zero. Finally, the outputs of both the object and global pathways f_{local_i} and f_{global} are concatenated along the channel axis and we again apply convolutional layers to obtain a merged feature representation. At this point, the features are concatenated either with the spatially replicated image caption embedding φ (if existent) or the sum of all one-hot vectors l_{onehot_i} along the channel axis, one more convolutional layer is applied, and the output is classified as either generated or real.

For the general training, we can utilize the same procedure that is used in the GAN architecture that is modified with our proposed approach. In our work we mostly use the StackGAN [Zhang et al., 2018a] and AttnGAN [Xu et al., 2018b] frameworks which use a modified objective function taking into consideration the additional conditional information and provided image captions. As such, our discriminator D and our generator G optimize the following objective function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{(x,c)\sim p_{\text{data}}}[logD(x,c)] + \mathbb{E}_{(z)\sim p_z,(c)\sim p_{\text{data}}}[log(1-D(G(z,c),c))],$$

where x is an image, c is the conditional information for this image (e.g. $label_i$, bounding boxes $bbox_i$, or an image caption φ), z is a randomly sampled noise vector used as input for G, and p_{data} is the true data distribution. Zhang et al. [2018a] and others use an additional technique called conditioning augmentation for the image captions which helps improve the training process and the quality of the generated images. In the experiments in which we use image captions (MS-COCO) we also make use of this technique.[1]

4.2.4 Evaluation and Analysis

For the evaluation, we aim to study the quality of the generated images with a particular focus on the generalization capabilities and the contribution of specific parts of our model, in both controllable and large-scale cases. Thus, in the following sections, we evaluate our approach on three different data sets: the Multi-MNIST data set, the CLEVR data set, and the MS-COCO data set.

Multi-MNIST

In our first experiment, we used the Multi-MNIST data set [Eslami et al., 2016] for testing the basic functionality of our proposed model. Using the implementation provided by Eslami et al. [2016], we created 50,000 images of resolution 64×64 px that contain exactly three normal-sized MNIST digits in non-overlapping locations on a black background.

[1] More detailed information about the implementation can be found in the Appendix.

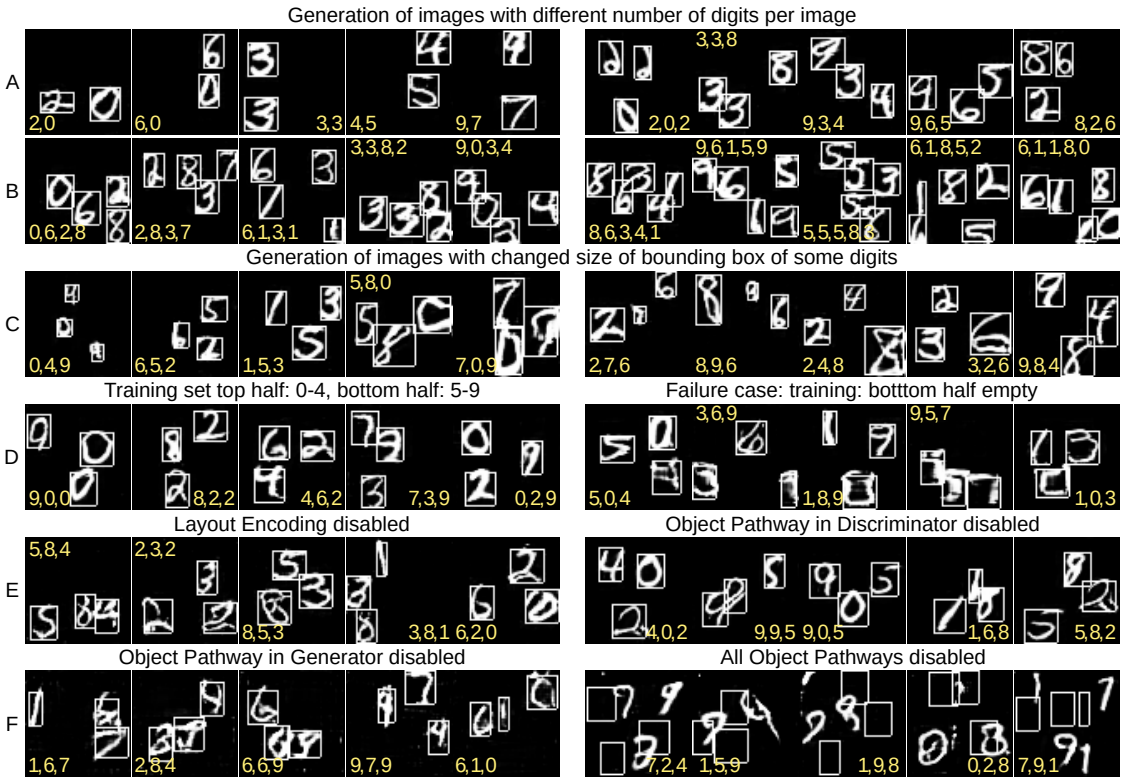


Figure 4.2: Multi-MNIST images generated by the model. Training included only images with three individual normal-sized digits. Highlighted bounding boxes and yellow ground truth for visualization.

As a first step, we tested whether our model can learn to generate digits at the specified locations and whether we can control the digit identity, the generated digit's size, and the number of generated digits per image. According to the results, we can control the location of individual digits, their identity, and their size, even though all training images contain exactly three digits in normal size. Figure 4.2 shows that we can control how many digits are generated within an image (rows A–B, for two to five digits) and various sizes of the bounding box (row C). As a second step, we created an additional Multi-MNIST data set in which all training images contain only digits 0–4 in the top half and only digits 5–9 in the bottom half of the image. For testing digits in the opposite half, we can see that the model is indeed capable of generalizing the position (row D, left), i.e. it can generate digits 0–4 in the bottom half of the image and digits 5–9 in the top half of the image. Nevertheless, we also observed that this does not always work perfectly, as the network sometimes alters digits towards the ones it has seen during training at the respective locations, e.g. producing a "4" more similar to a "9" if in bottom half of the image, or generating a "7" more similar to a "1" if in top half of the image.

As a next step, we created a Multi-MNIST data set with images that only contain digits in the top half of the image, while the bottom half is always empty. We can see (Figure 4.2, row D, right) that the resulting model is not able to generate digits in the bottom half of the image (see Figure B.1 in the Appendix for more

details on this). Controlling for the location still works, i.e. bounding boxes are filled with "something", but the digit identity is not clearly recognizable. Thus, the model is able to control both the object identity and the object location within an image and can generalize to novel object locations to some extent.

To test the impact of our model extensions, i.e. the object pathway in both the generator and the discriminator as well as the layout encoding, we performed ablation studies on the previously created Multi-MNIST data set with three digits at random locations. We first disabled the use of the layout encoding in the generator and left the rest of the model unchanged. In the results (Figure 4.2, row E, left), we can see that, overall, both the digit identity and the digit locations are still correct, but minor imperfections can be observed within various images. This is most likely due to the fact that the global pathway of the generator has no information about the digit identity and location until its features get merged with the object pathway. As a next test, we disabled the object pathway of the discriminator and left the rest of the model unmodified. Again, we see (row E, right) that we can still control the digit location, although, again, minor imperfections are visible. More strikingly, we have a noticeably higher error rate in the digit identity, i.e. the wrong digit is generated at a given location, most likely due to the fact that there is not object pathway in the discriminator controlling the object identity at the various locations. In comparison, the imperfections are different when only the object pathway of the generator is disabled (row F, left). The layout encoding and the feedback of the discriminator seem to be enough to still produce the digits in the correct image location, but the digit identity is often incorrect or not recognizable at all. Finally, we tested disabling the object pathway in both the discriminator and the generator (see row F, right). This leads to a loss of control of both image location as well as identity and sometimes even results in images with more or fewer than three digits per image. This shows that only the layout encoding, without any of the object pathways, is not enough to control the digit identity and location. Overall, these results indicate that we do indeed need both the layout encoding, for a better integration of the global and object pathways, and the object pathways in both the discriminator and the generator, for optimal results.

CLEVR

In our second experiment we used more complex images containing multiple objects of different colors and shapes. The goal of this experiment was to evaluate the generalization ability of our object pathway across different object characteristics. For this, we performed tests similar to [Raj et al., 2017], albeit on the more complex CLEVR data set [Johnson et al., 2017]. In the CLEVR data set objects are characterized by multiple properties, in our case the shape, the color, and the size. Based on the implementation provided by Johnson et al. [2017], we rendered 25,000 images with a resolution of 64×64 pixels containing $2 - 4$ objects per image. The label for a given bounding box of an object is the object shape and color (both encoded as one-hot encoding and then concatenated), while the object size is specified through the height and width of the bounding box.

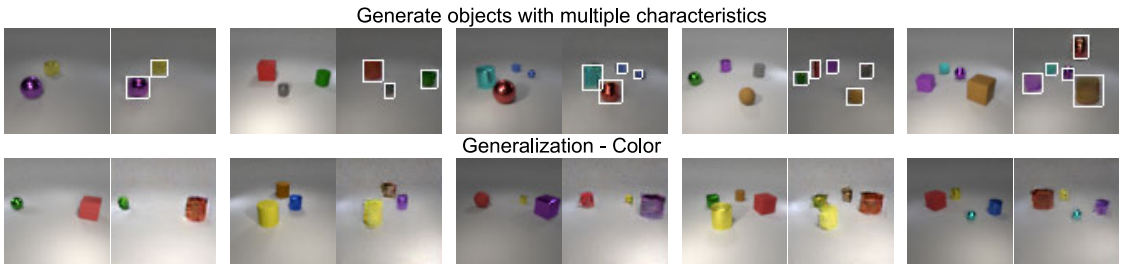


Figure 4.3: Images from the CLEVR data set. The left image of each pair shows the rendered image according to specific attributes. The right image of each pair is the image generated by our model.

Similar to the first experiment, we tested our model for controlling the object characteristics, size, and location. In the first row of Figure 4.3 we present the results of the trained model, where the left image of each pair shows the originally rendered one, while the right image was generated by our model. We can confirm that the model can control both the location and the objects' shape and color characteristics. The model can also generate images containing an arbitrary number of objects (forth and fifths pair), even though a maximum of four objects per image was seen during training.

The CLEVR data set offers a split specifically intended to test the generalization capability of a model, in which cylinders can be either red, green, purple, or cyan and cubes can be either gray, blue, brown, or yellow during training, while spheres can have any of these colors. During testing, the colors between cylinders and cubes are reversed. Based on these restrictions, we created a second data set of 25,000 training images for testing our model. Results of the test are shown in the second row of Figure 4.3 (again, left image of each pair shows the originally rendered one, while the right image was generated by our model). We can see that the color transfer to novel shape-color combinations takes place, but, similarly to the Multi-MNIST results, we can see some artifacts, where e.g. some cubes look a bit more like cylinders and vice versa. Overall, the CLEVR experiment confirms the indication that our model can control object characteristics (provided through labels) and object locations (provided through bounding boxes) and can generalize to novel object locations, novel amounts of objects per image, and novel object characteristic combinations within reasonable boundaries.

MS-COCO

For our final experiment, we used the MS-COCO data set [Lin et al., 2014] to evaluate our model on natural images of complex scenes. In order to keep our evaluation comparable to previous work, we used the 2014 train/test split consisting of roughly 80,000 training and 40,000 test images and rescaled the images to a resolution of 256×256 px. At train-time, we used the bounding boxes and object labels of the three largest objects within an image, i.e. we used zero to three bounding boxes per image. Similarly to work by Johnson et al. [2018] we only

considered objects that cover at least 2% of the image for the bounding boxes. To evaluate our results quantitatively, we computed both the Inception Score (IS, larger is better), which tries to evaluate how recognizable and diverse objects within images are [Salimans et al., 2016], as well as the Fréchet Inception Distance (FID, smaller is better), which compares the statistics of generated images with real images [Heusel et al., 2017]. As a qualitative evaluation, we generated images that contain more than one object, and checked, whether the bounding boxes can control the object placement. We tested our approach with two commonly used architectures for text-to-image synthesis, namely the StackGAN [Zhang et al., 2017b] and the AttnGAN [Xu et al., 2018b], and compared the images generated by these and our models.

In the StackGAN, the training process is divided into two steps: first, it learns a generator for images with a resolution of 64×64 px based on the image captions, and second, it trains a second generator, which uses the smaller images (64×64 px) from the first generator and the image caption as input to generate images with a resolution of 256×256 px. Here, we added the object pathways and the layout encoding at the beginning of both the first generator and the second generator and used the object pathway in both discriminators. The other parts of StackGAN architecture and all hyperparameters remain the same as in the original training procedure for the MS-COCO data set. We trained the model three times from scratch and randomly sampled 3 times 30,000 image captions from the test set for each model. We then calculated the IS and FID values on each of the nine samples of 30,000 generated images and report the averaged values. As presented in Table 4.1, our StackGAN with added object pathways outperforms the original StackGAN both on the IS and the FID, increasing the IS from 10.62 to 12.12 and decreasing the FID from 74.05 to 55.30. Note, however, that this might also be due to the additional information our model is provided with as it receives up to three bounding boxes and respective bounding box labels per image in addition to the image caption.

We also extended the AttnGAN by Xu et al. [2018b], the current state-of-the-art model on the MS-COCO data set (based on the Inception Score), with our object pathway to evaluate its impact on a different model. As opposed to the StackGAN, the AttnGAN consists of only one model which is trained end-to-end on the image captions by making use of multiple, intermediate, discriminators. Three discriminators judge the output of the generator at an image resolution of 64×64, 128×128, and 256×256 px. Through this, the image generation process is guided at multiple levels, which helps during the training process. Additionally, the AttnGAN implements an attention technique through which the networks focus on specific areas of the image for specific words in the image caption and adds an additional loss that checks if the image depicts the content as described by the image caption. There, in the same way as for the StackGAN, we added our object pathway at the beginning of the generator as well as to the discriminator that judges the generator outputs at a resolution of 64×64 px. All other discriminators, the higher layers of the generator, and all other hyperparameters and training details stay unchanged. Table 4.1 shows that adding the object pathway to the

Model	Resolution	IS ↑	FID ↓
GAN-INT-CLS [Reed et al., 2016b]	64×64	7.88 ± 0.07	60.62
StackGAN-V2 [Zhang et al., 2018a]	256×256	8.30 ± 0.10	81.59
StackGAN [Zhang et al., 2018a]	256×256	8.45 ± 0.03[1]	74.05
PPGN [Nguyen et al., 2017]	227×227	9.58 ± 0.21	
ChatPainter [Sharma et al., 2018]	256×256	9.74 ± 0.02	
Semantic Layout [Hong et al., 2018b]	128×128	11.46 ± 0.09[2]	
HDGan [Zhang et al., 2018d]	256×256	11.86 ± 0.18	71.27 ± 0.12[3]
AttnGAN [Xu et al., 2018b]	256×256	23.61 ± 0.21[4]	33.10 ± 0.11[3]
StackGAN + Object Pathways (Ours)[5]	256×256	12.12 ± 0.31	55.30 ± 1.78
AttnGAN + Object Pathways (Ours)	256×256	24.76 ± 0.43	33.35 ± 1.15

[1] Recently updated to 10.62 ± 0.19 in its source code.
[2] When using the ground truth bounding boxes at test time (as we do) the IS increases to 11.94 ± 0.09.
[3] FID score was calculated with samples generated with the pretrained model provided by the authors.
[4] The authors report a "best" value of 25.89 ± 0.47, but when calculating the IS with the pretrained model provided by the authors we only obtain an IS of 23.61. Other researchers on the authors' Github website report a similar value for the pretrained model.
[5] We use the updated source code (IS of 10.62) as our baseline model.

Table 4.1: Comparison of the Inception Score (IS) and Fréchet Inception Distance (FID) on the MS-COCO data set for different models. Note: the IS and FID values of our models are not necessarily directly comparable to the other models, since our model gets at test time, in addition to the image caption, up to three bounding boxes and their respective object labels as input.

AttnGAN increases the IS of our baseline model (the pretrained model provided by the authors) from 23.61 to 24.76, while the FID is roughly the same as for the baseline model.

To evaluate whether the StackGAN model equipped with an object pathway (StackGAN+OP) actually generates objects at the given positions we generated images that contain multiple objects and inspected them visually. Figure 4.4 shows some example images, more results can be seen in the Appendix in Figures B.2 and B.4. We can observe that the StackGAN+OP indeed generates images in which the objects are at appropriate locations. In order to more closely inspect our global and object pathways, we can also disable them during the image generation process. Figure 4.5 shows additional examples, in which we generate the same image with either the global or the object pathway disabled during the generation process. Row C of Figure 4.5 shows images in which the object pathway was disabled and, indeed, we observe that the images contain mostly background information and objects at the location of the bounding boxes are either not present or of much less

Figure 4.4: Examples of images generated from the given caption from the MS-COCO data set. *A)* shows the original images and the respective image captions, *B)* shows images generated by our StackGAN+OP (with the corresponding bounding boxes for visualization), and *C)* shows images generated by the original StackGAN [Zhang et al., 2017b][2].

detail than when the object pathway is enabled. Conversely, row D of Figure 4.5 shows images which were generated when the global pathway was disabled. As expected, areas outside of the bounding boxes are empty, but we also observe that the bounding boxes indeed contain images that resemble the appropriate objects. These results indicate, as in the previous experiments, that the global pathway does indeed model holistic image features, while the object pathway focuses on specific, individual objects.

When we add the object pathway to the AttnGAN (AttnGAN + OP) we can observe similar results[3]. Again, we are able to control the location and identity of objects through the object pathway, however, we observe that the AttnGAN+OP, as well as the AttnGAN in general, tends to place objects corresponding to specific features at many locations throughout the image. For example, if the caption contains the word "traffic light" the AttnGAN tends to place objects similar to traffic lights throughout the whole image. Since our model only focuses on generating objects at given locations, while not enforcing that these objects *only* occur at these locations, this behavior leads to the result that the AttnGAN+OP generates desired objects at the desired locations, but might also place the same object at other locations within the image. Note, however, that we only added the object pathway to the lowest generator and discriminator and that we might gain even more control over the object location by introducing object pathways to the higher generators and discriminators, too.

In order to further evaluate the quality of the generations, we ran an object detection test on the generated images using a pretrained YOLOv3 network [Redmon and Farhadi, 2018]. Here, the goal is to measure how often an object detection framework, which was trained on MS-COSO as well, can detect a specified object

[2]Generated with the model from: https://github.com/hanzhanggit/StackGAN-Pytorch

[3]Examples of images generated by the AttnGAN+OP can be seen in the Appendix in Figures B.3 and B.5.

at a specified location.[4] The results confirm the previously made observations: For both the StackGAN and the AttnGAN the object pathway seems to improve the image quality, since YOLOv3 detects a given object more often correctly when the images are generated with an object pathway as opposed to images generated with the baseline models. The StackGAN generates objects at the given bounding box, resulting in an Intersection over Union (IoU) of greater than 0.3 for all tested labels and greater than 0.5 for 86.7% of the tested labels. In contrast, the AttnGAN tends to place salient object features throughout the image, which leads to an even higher detection rate by the YOLOv3 network, but a smaller average IoU (only 53.3% of the labels achieve an IoU greater than 0.3). Overall, our experiments on the MS-COCO data set indicate that it is possible to add our object pathway to pre-existing GAN models without having to change the overall model architecture or training process. Adding the object pathway provides us with more control over the image generation process and can, in some cases, increase the quality of the generated images as measured via the IS or FID.

Discussion

Our experiments indicate that we do indeed get additional control over the image generation process through the introduction of object pathways in GANs. This enables us to control the identity and location of multiple objects within a given image based on bounding boxes and thereby facilitates the generation of more complex scenes. We further find that the division of work on a global and object pathway seems to improve the image quality both subjectively and based on quantitative metrics such as the Inception Score and the Fréchet Inception Distance.

The results further indicate that the focus on global image statistics by the global pathway and the more fine-grained attention to detail of specific objects by the object pathway works well. This is visualized for example in rows C and D of Figure 4.5. The global pathway (row C) generates features for the general image layout and background but does not provide sufficient details for individual objects. The object pathway (row D), on the other hand, focuses entirely on the individual objects and generates features specifically for a given object at a given location. While this is the desired behavior of our model it can also lead to sub-optimal images if there are not bounding boxes for objects that should be present within the image. This can often be the case if the foreground object is too small (in our case less than 2% of the total image) and is therefore not specifically labeled. In this case, the objects are sometimes not modeled in the image at all, despite being prominent in the respective image caption, since the object pathway does not generate any features. We can observe this, for example, in images described as "*many sheep are standing on the grass*", where the individual sheep are too small to warrant a bounding box. In this case, our model will often only generate an image depicting grass and other background details, while not containing any sheep at all.

Another weakness is that bounding boxes that overlap too much (empirically

[4]See Appendix for more details on the procedure and the exact results.

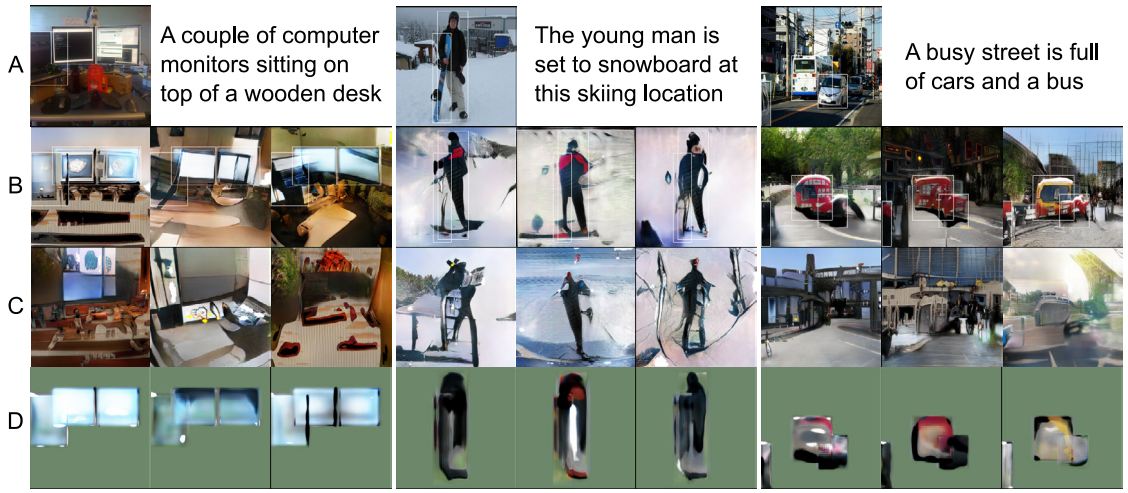


Figure 4.5: Examples of images generated from the given caption from the MS-COCO data set. *A)* shows the original images and the respective image captions, *B)* shows images generated by our StackGAN+OP (with the corresponding bounding boxes for visualization) with the object pathway enabled, *C)* shows images generated by the our StackGAN+OP when the object pathway is disabled, and *D)* shows images generated by the our StackGAN+OP when the global pathway is disabled.

an overlap of more than roughly 30%) also often lead to sub-optimal objects at that location. Especially in the overlapping section of bounding boxes we often observe local inconsistencies or failures. This might be the result of our merging of the different features within the object pathway since they are simply added to each other at overlapping areas. A more sophisticated merging procedure could potentially alleviate this problem.Another approach would be to additionally enhance the bounding box layout by predicting the specific object shape within each bounding box, as done for example by Hong et al. [2018b].

Finally, currently our model does not generate the bounding boxes and labels automatically. Instead, they have to be provided at test time which somewhat limits the usability for unsupervised image generation. However, even when using ground truth bounding boxes, our models still outperform other current approaches that are tested with ground truth bounding boxes (e.g. Hong et al. [2018b]) based on the IS and FID. This is even without the additional need of learning to specify the shape within each bounding box as done by Hong et al. [2018b]. In the future, this limitation can be avoided by extracting the relevant bounding boxes and labels directly from the image caption, as it is done for example by Hong et al. [2018b], Xu et al. [2018a], and Tan et al. [2019].

4.2.5 Conclusion

With the goal of understanding how to gain more control over the image generation process in GANs, we introduced the concept of an additional object pathway. Such a mechanism for differentiating between a scene representation and object

representations allows us to control the identity, location, and size of arbitrarily many objects within an image, as long as the objects do not overlap too strongly. In parallel, a global pathway, similar to a standard GAN, focuses on the general scene layout and generates holistic image features. The object pathway, on the other hand, gets as input an object label and uses this to generate features specifically for this object which are then placed at the location given by a bounding box The object pathway is applied iteratively for each object at each given location and as such, we obtain a representation of individual objects at individual locations and of the general image layout (background, etc.) as a whole. The features generated by the object and global pathway are then concatenated and are used to generate the final image output. Our tests on synthetic and real-world data sets suggest that the object pathway is an extension that can be added to common GAN architectures without much change to the original architecture and can, along with more fine-grained control over the image layout, also lead to better image quality.

4.3 Evaluating Compositional Representations

This section presents our work *Semantic object accuracy for generative text-to-image synthesis* by Tobias Hinz, Stefan Heinrich, and Stefan Wermter published in 2020 in *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

4.3.1 Introduction

Generative adversarial networks (GANs) [Goodfellow et al., 2014] are capable of generating realistic-looking images that adhere to characteristics described in a textual manner, e.g. an image caption. For this, most networks are conditioned on an embedding of the textual description. Often, the textual description is used on multiple levels of resolution, e.g. first to obtain a course layout of the image at lower levels and then to improve the details of the image on higher resolutions. This approach has led to good results on simple, well-structured data sets containing a specific class of objects (e.g. faces, birds, or flowers) at the image center.

Once images and textual descriptions become more complex, e.g. by containing more than one object and having a large variety in backgrounds and scenery settings, the image quality drops drastically. This is likely because, until recently, almost all approaches only condition on an embedding of the complete textual description, without paying attention to individual objects. Recent approaches have started to tackle this by either relying on specific scene layouts [Johnson et al., 2018] or by explicitly focusing on individual objects [Hinz et al., 2019; Li et al., 2019d]. In this work, we extend this approach by additionally focusing specifically on salient objects within the generated image. However, generating complex scenes containing multiple objects from a variety of classes is still a challenging problem.

The most commonly used evaluation metrics for GANs, the Inception Score (IS) [Salimans et al., 2016] and the Fréchet Inception Distance (FID) [Heusel et al.,

2017], are not designed to evaluate images that contain multiple objects and depict complex scenes. In fact, both of these metrics depend on an image classifier (the Inception-Net), which is pre-trained on ImageNet, a data set whose images almost always contain only a single object at the image center. They also do not evaluate the consistency between image description and generated image and, therefore, can not evaluate whether a model generates images that actually depict what is described in the caption. Even evaluation metrics specifically designed for text-to-image synthesis evaluation such as the R-precision [Xu et al., 2018b] often fail to evaluate more detailed aspects of an image, such as the quality of individual objects.

As such, our contributions are twofold: first, we introduce a novel GAN architecture called *OP-GAN* that focuses specifically on individual objects while simultaneously generating a background that fits with the overall image description. Our approach relies on an object pathway similar to Hinz et al. [2019], which iteratively attends to all objects that need to be generated given the current image description. In parallel, a global pathway generates the background features which later on get merged with the object features. Second, we introduce an evaluation metric specifically for text-to-image synthesis tasks which we call *Semantic Object Accuracy* (SOA). In contrast to most current evaluation metrics, our metric focuses on individual objects and parts of an image and also takes the caption into consideration when evaluating an image. Image descriptions often explicitly or implicitly mention what kind of objects are seen in an image, e.g. an image described by the caption *"a person holding a cell phone"* should depict both a person and a cell phone. To evaluate this, we sample all image captions from the COCO validation set that explicitly mention one of the 80 main object categories (e.e. "person", "dog", "car", etc.) and use them to generate images. We then use a pre-trained object detector [Redmon and Farhadi, 2018] and check whether it detects the explicitly mentioned objects within the generated images. We perform a user study over several current text-to-image models and show that SOA is highly compatible with human evaluation whereas other metrics, such as the Inception Score, are not.

We evaluate several variations of our proposed model as well as several state-of-the-art approaches that provide pre-trained models. Our results show that current architectures are not able to generate images that contain objects of the same quality as the original images. While some models already achieve results close to or better than real images on scores such as the IS and R-precision, none of the models comes close to generating images that achieve SOA scores close to the real images. However, our results and user study also show that models that attend to individual objects in one way or another tend to perform better than models, which only focus on global image semantics.

4.3.2 Related Work

Modern architectures are able to synthesize realistic, high-resolution images of many domains. In order to generate images of high resolution many GAN [Goodfellow

et al., 2014] architectures use multiple discriminators at various resolutions [Zhang et al., 2018a]. Additionally, most GAN architectures use some form of attention for improved image synthesis [Xu et al., 2018b] as well as matching aware discriminators [Reed et al., 2016b] which identify whether images correspond to a given textual description.

Originally, most GAN approaches for text-to-image synthesis encoded the textual description into a single vector which was used as a condition in a conditional GAN (cGAN) [Reed et al., 2016b; Zhang et al., 2018a]. However, this faces limitations when the image content becomes more complex as e.g. in the COCO data set [Lin et al., 2014]. As a result, many approaches now use attention mechanisms to attend to specific words of the sentence [Xu et al., 2018b], use intermediate representations such as scene layouts [Johnson et al., 2018], condition on additional information such as object bounding boxes [Hinz et al., 2019] or perform interactive image refinement [Sharma et al., 2018]. Other approaches generate images directly from semantic layouts without additional textual input [Karacan et al., 2016; Park et al., 2019]or perform a translation from text to images and back [Sah et al., 2018; Qiao et al., 2019b].

Direct Text-to-Image Synthesis Approaches that do not use intermediate representations such as scene layouts use only the image caption as conditional input. Reed et al. [2016b] use a GAN to generate images from captions directly and without any attention mechanism. Captions are embedded and used as conditioning vector and they introduce the widely adopted matching aware discriminator. The matching aware discriminator is trained to distinguish between real and matching caption-image pairs ("real"), real but mismatching caption-image pairs ("fake"), and matching captions with generated images ("fake"). Cha et al. [2019] modify the sampling procedure during training to obtain a curriculum of mismatching caption-image pairs and introduce an auxiliary classifier that specifically predicts the semantic consistency of a given caption-image pair. Zhang et al. [2017b, 2018a] use multiple generators and discriminators and are one of the first ones to achieve good image quality at resolutions of 256×256 on complex data sets. Zhang et al. [2018d] have a similar architecture as Zhang et al. [2017b] with multiple discriminators but only use one generator while Huang et al. [2019b] generate realistic high-resolution images from text with a single discriminator and generator.

Xu et al. [2018b] extend Zhang et al. [2018a] and are the first ones to introduce an attention mechanism to the text-to-image synthesis task with GANs. The attention mechanism attends to specific words in the caption and conditions different image regions on different words to improve the image quality. Yin et al. [2019] extend this and also consider semantics from the text description during the generation process. Zhu et al. [2019] introduce a dynamic memory part that selects "bad" parts of the initial image and tries to refine them based on the most relevant words. Li et al. [2019a] refine the attention module by having spatial and channel-wise word-level attention and introduce a word-level discriminator to provide fine-grained feedback based on individual words and image regions. Qiao et al. [2019a] decompose the text-to-image process into three distinct phases by first learning a prior over the text-image space, then sampling from this prior, and lastly using the prior to

generate the image.

Text-to-Image Synthesis with Layouts When using more complex data sets that contain multiple objects per image, generating an image directly becomes difficult. Therefore, many approaches use additional information such as bounding boxes for objects or intermediate representations such as scene graphs or scene layouts which can be generated automatically [Li et al., 2019c; Jyothi et al., 2019; Li et al., 2019b]. Reed et al. [2016c] and Reed et al. [2016a] build on Reed et al. [2016b] by additionally conditioning the generator on bounding boxes or keypoints of relevant objects. Raj et al. [2017] decomposition textual descriptions into basic visual primitives to generate images in a compositional manner. Johnson et al. [2018] introduce the concept of generating a scene graph based on a caption. This scene graph is then used to generate an image layout and finally the image. Similar to Johnson et al. [2018], Hong et al. [2018b] use the caption to infer a scene layout which is used to generate images. Liu et al. [2019] predict convolution kernels conditioned on the semantic layout, making it possible to control the generation process based on semantic information at different locations.

Given a coarse image layout (bounding boxes and object labels) Zhao et al. [2019] generate images by disentangling each object into a specified part (e.g. object label) and unspecified part (appearance). Hinz et al. [2019] generate images conditioned on bounding boxes for the individual foreground objects by introducing an object pathway that generates individual objects. Li et al. [2019d] update the grid-based attention mechanism Xu et al. [2018b] by combining attention with scene layouts. Additionally, an object discriminator is introduced which focuses on individual objects and provides feedback whether the object is at the right location. Huang et al. [2019a] refine the grid-based attention mechanism between word phrases and specific image regions of various sizes based on an initial set of bounding boxes. Sun and Wu [2019] introduce a new feature normalization method and fine-grained mask maps to generate visually different images from a given layout. Li et al. [2019f] generate images from scene graphs and allow the model to crop objects from other images to paste them into the generated image. Vo and Sugimoto [2020] generate a visual-relation scene layout based on the caption. For this, they introduce a dedicated module which generates bounding boxes for objects at a given caption in order to condition the network during the image generation process.

Semantic Image Manipulation Finally, there are methods that allow humans to directly describe the image in an iterative process or that allow for direct semantic manipulation of images. Sharma et al. [2018] condition generation process on a dialogue describing the image instead of a single caption. Hong et al. [2018a] facilitate semantic image manipulation by allowing users to modify image layouts which are then used to generate images. Lee et al. [2018] allow users to input object instance masks into an existing image represented by a semantic layout. El-Nouby et al. [2019] generate images iteratively from consecutive textual commands, Cheng et al. [2020] provide interactive image editing based on a current image and instructions on how to update the image, and Li et al. [2019e] generate individual images for a sequence of sentences. Mittal et al. [2019] do interactive image generation but

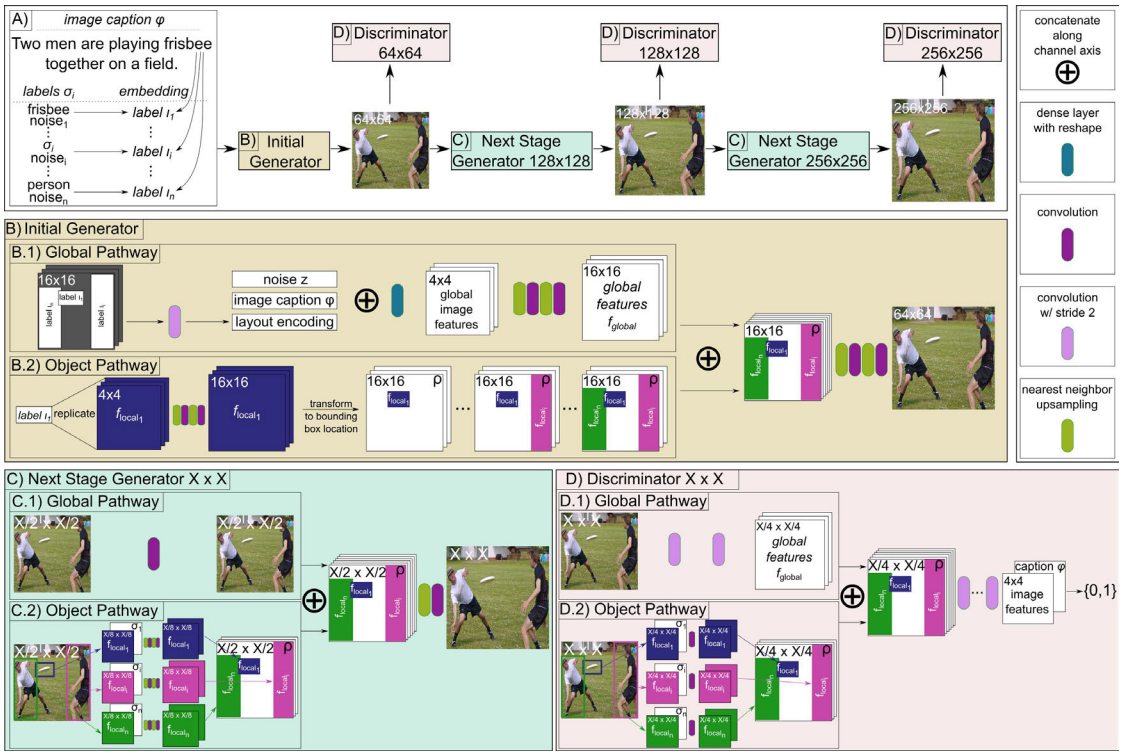


Figure 4.6: Overview of our model architecture called *OP-GAN*. The top row shows a high-level summary of our architecture, while the bottom two rows show details of the individual generators and discriminators.

do not use text as direct input but instead update a scene graph from text over the course of the interaction. Nam et al. [2018]; Zhou et al. [2019], and Yu et al. [2019] modify visual attributes of individual objects in an image while leaving text irrelevant parts of the image unchanged.

4.3.3 Approach

A traditional generative adversarial network (GAN) [Goodfellow et al., 2014] consists of two networks: a generator G which generates new data points from randomly sampled inputs, and a discriminator D which tries to distinguish between generated and real data samples. In conditional GANs (cGANs) [Mirza and Osindero, 2014] both the discriminator and the generator are conditioned on additional information, e.g. a class label or textual information. This has been shown to improve performance and leads to more control over the data generating process. For a conventional cGAN with generator G, discriminator D, condition c (e.g. a class label), data point x, and a randomly sampled noise vector z the training objective V is:

$$\begin{aligned}\min_G \max_D V(D,G) = \mathbb{E}_{(x,c)\sim p_{\text{data}}}[log\ D(x,c)]+ \\ \mathbb{E}_{(z)\sim p_z,(c)\sim p_{\text{data}}}[log(1-D(G(z,c),c))].\end{aligned} \tag{4.1}$$

We use the AttnGAN [Xu et al., 2018b] as our baseline architecture and add our object-centric modifications to it. The AttnGAN is a conditional GAN for text-to-image synthesis that uses attention and a novel additional loss to improve the quality of the generated images. It consists of a generator and three discriminators as shown in the top row of Figure 4.6. Attention is used such that different words of the caption have more or less influence on different regions of the image. This means that, for example, the word "sky" has more influence on the generation of the top half of the image than the word "grass" even if both words are present in the image caption.

Xu et al. [2018b] also introduce the Deep Attentional Multimodal Similarity Model (DAMSM) which computes the similarity between images and captions. This DAMSM is used during training to provide additional, fine-grained feedback to the generator about how well the generated image matches its caption. We adapt the AttnGAN architecture with multiple object pathways which are learned end-to-end in both the discriminator and the generator, see B and C in Figure 4.6.

These object pathways are conditioned on individual object labels (e.g. "person", "car", etc.) and the same object pathway is applied multiple times at a given image resolution at different locations and for different objects. This is similar to the approach introduced by Hinz et al. [2019]. However, Hinz et al. [2019] only use one object pathway in the generator at a small resolution and only one discriminator was equipped with an object pathway. In our approach, the generator contains three object pathways at various resolutions (16×16, 64×64, and 128×128) to further refine object features at higher resolutions and each of our three discriminators is equipped with its own object pathway, see D in Figure 4.6.

For a given image caption φ we have several objects which are associated with this caption and which we represent with one-hot vectors $\sigma_i, i = 1...n$ (e.g. $\sigma_0 =$ person, $\sigma_1 =$ car, etc.). Each object pathway at a given resolution is applied iteratively for each of the objects σ_i. The location is determined by a bounding box describing the object's location and size. Each object pathway starts with an "empty" zero-tensor ρ and the features that are generated (generator) or extracted (discriminator) are added onto ρ at the location of the specific object's bounding box. After the object pathway has processed each object, ρ contains features at each object location and is zero everywhere else.

For the generator, we first concatenate the image caption's embedding φ, the one-hot label σ_i, and a randomly sampled noise vector z. We use this concatenated vector to obtain the final conditioning label ι_i for the current object σ_i:

$$\iota_i = \mathrm{F}(\varphi, z, \sigma_i), \tag{4.2}$$

where F is a fully connected layer followed by a non-linearity (A in Figure 4.6).

The generator's first object pathway (*B.2* in Figure 4.6) takes this conditioning label ι_i and uses it to generate features for the given object at a spatial resolution of 16×16. The features are then transformed onto ρ into the location of the respective bounding box with a spatial transformer network (STN) [Jaderberg et al., 2015]. This procedure is repeated for each object σ_i associated with the given caption φ.

The global pathway in the first generator also gets the locations and labels ι_i for the individual objects. It spatially replicates these labels at the locations of the respective bounding boxes and then applies convolutional layers to the resulting layout to obtain a layout encoding (*B.1* in Figure 4.6). This layout encoding, the image caption φ, and the noise vector z are used to generate coarse features for the image at a low resolution.

At higher levels in the generator, the object pathways are conditioned on the object features of the current object and the one-hot label σ_i for that object (*C.2* in Figure 4.6). For this, we again use an STN to extract the features at the bounding box location of the object σ_i and resize the features to a spatial resolution of 16×16 (second object pathway) or 32×32 (third object pathway). We obtain a conditioning label in the same manner as for the first object pathway (Equation 4.2), replicate it spatially to the same dimension as the extracted object features, and concatenate it with the object features along the channel axis. Following this, we apply multiple convolutional layers and upsampling to update the features of the given object. Finally, as in the first object pathway, we use an STN to transform the features into the bounding box location and add them onto ρ. The global pathway in the higher layers (*C.1* in Figure 4.6) stays unchanged from the baseline architecture [Xu et al., 2018b].

Our final loss function for the generator is the same as in the original AttnGAN and consists of an unconditional, a conditional, and a caption-image matching part. The unconditional loss is

$$\mathcal{L}_G^{\text{uncon}} = -\mathbb{E}_{(\hat{x})\sim p_G}[log\ D(\hat{x}))], \tag{4.3}$$

the conditional loss is

$$\mathcal{L}_G^{\text{con}} = -\mathbb{E}_{(\hat{x})\sim p_G,(c)\sim p_{\text{data}}}[log\ D(\hat{x}, c))], \tag{4.4}$$

and the caption-image matching loss is $\mathcal{L}_G^{\text{DAMSM}}$ [Xu et al., 2018b] which measures text-image similarity at the word level and is calculated with the pre-trained models provided by Xu et al. [2018b]. The complete loss for the generator then is:

$$\mathcal{L}_G = \mathcal{L}_G^{\text{uncon}} + \mathcal{L}_G^{\text{con}} + \lambda\mathcal{L}_G^{\text{DAMSM}}, \tag{4.5}$$

where we set $\lambda = 50$ as in the original implementation.

As in our baseline architecture, we employ three discriminators at three spatial resolutions: 64×64, 128×128, and 256×256. Each discriminator possesses a global and an object pathway which extract features in parallel (D in Figure 4.6). In the object pathway we use an STN to extract the features of object σ_i and concatenate them with the one-hot vector σ_i describing the object. The object pathway then applies multiple convolutional layers before adding the extracted features onto ρ at the location of the bounding box.

The global pathway in each of the discriminators works on the full input image and applies convolutional layers with stride two to decrease the spatial resolution (*D.1*). Once the spatial resolution reaches that of the tensor ρ we concatenate the

two tensors (full image features and object features ρ) along the channel axis and use convolutional layers with stride two to further reduce the spatial dimension until we reach a resolution of 4×4.

We calculate both a conditional (image and image caption) and an unconditional (only image) loss for each of the discriminators. The conditional input c during training consists of the image caption embedding φ and the information about objects σ_i (bounding boxes and object labels) associated with the image x, i.e. $c = \{\varphi, \sigma_i\}$. In the unconditional case the discriminators are trained to classify images as real or generated without any influence of the image caption by minimizing the following loss:

$$\mathcal{L}_{D_i}^{\text{uncon}} = -\mathbb{E}_{(x)\sim p_{\text{data}}}[log\ D(x)] - \mathbb{E}_{(\hat{x})\sim p_G}[log(1 - D(\hat{x}))]. \tag{4.6}$$

In order to optimize the conditional loss we concatenate the extracted features with the image caption embedding φ along the channel axis and minimize

$$\begin{aligned} \mathcal{L}_{D_i}^{\text{con}} &= -\mathbb{E}_{(x,c)\sim p_{\text{data}}}[log\ D(x, c)] \\ -\mathbb{E}_{(\hat{x})\sim p_G, (c)\sim p_{\text{data}}}&[log(1 - D(\hat{x}, c))]. \end{aligned} \tag{4.7}$$

for each discriminator. Finally, to specifically train the discriminators to check for caption-image consistency we use the matching aware discriminator loss [Reed et al., 2016b] with mismatching caption-image pairs and minimize

$$\mathcal{L}_{D_i}^{\text{cls}} = -\mathbb{E}_{(x,\sigma)\sim p_{\text{data}}, (\varphi)\sim p_{\text{data}}}[log(1 - D(x, c))], \tag{4.8}$$

where image x and caption φ are sampled individually and randomly from the data distribution and are, therefore, unlikely to align.

We introduce an additional loss term similar to the matching aware discriminator loss $V_{\text{cls}}(D)$ which works on individual objects. Instead of using mismatching image-caption pairs, we use correct image-caption pairs, but with incorrect bounding boxes and minimize:

$$\mathcal{L}_{D_i}^{\text{obj}} = -\mathbb{E}_{(x,\varphi)\sim p_{\text{data}}, (\sigma)\sim p_{\text{data}}}[log(1 - D(x, c))]. \tag{4.9}$$

Thus, the complete objective we minimize for each individual discriminator is:

$$\mathcal{L}_{D_i} = \mathcal{L}_{D_i}^{\text{uncon}} + \mathcal{L}_{D_i}^{\text{con}} + \mathcal{L}_{D_i}^{\text{cls}} + \mathcal{L}_{D_i}^{\text{obj}}. \tag{4.10}$$

We leave all other training parameters as in the original implementation [Xu et al., 2018b] and the training procedure itself also stays the same.

4.3.4 Evaluation of Text-to-Image Models

Quantitatively evaluating generative models is difficult [Theis et al., 2016]. While there are several evaluation metrics that are commonly used to evaluate GANs, many of them have known weaknesses and are not designed specifically for text-to-image synthesis tasks. In the following, we first discuss some of the common evaluation metrics for GANs, their weaknesses, and why they might be inadequate for evaluating text-to-image synthesis models. Following this, we introduce our novel evaluation metric, Semantic Object Accuracy (SOA), and describe how it can be used to evaluate text-to-image models in more detail.

Current Evaluation Metrics

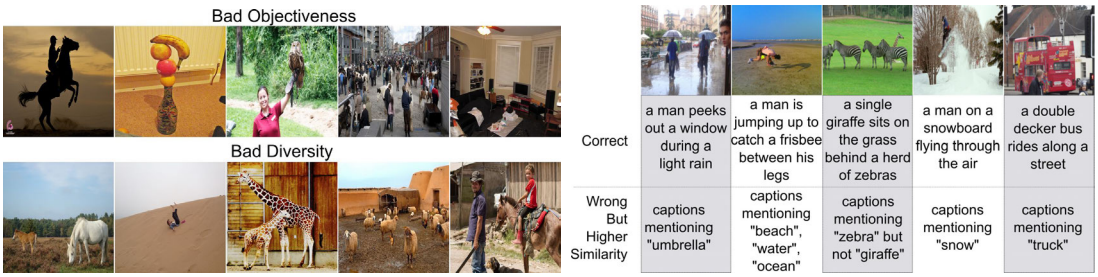
Inception Score and Fréchet Inception Distance Most GAN approaches are trained on relatively simple images which only contain one object at the center (e.g. ImageNet, CelebA, etc). These methods are evaluated with metrics such as the Inception Score (IS) [Salimans et al., 2016] and Fréchet Inception Distance (FID) [Heusel et al., 2017], which use an Inception-Net usually pre-trained on ImageNet. The IS evaluates roughly how distinctive an object in each image is (i.e. ideally the classification layer of the Inception-Net has small entropy) and how many different objects the GAN generates overall (i.e. high entropy in the output of different images). The FID measures how similar generated images are to a control set of images, usually the validation set by calculating the distance in feature space between generated and real images. Consequently, the IS should be as high as possible, while the FID should be as small as possible.

Both evaluation metrics have known weaknesses [Borji, 2019; Barratt and Sharma, 2018]. For example, the IS does not measure the similarity between objects of the same class, so a network that only generates one "perfect" sample for each class can achieve a very good IS despite showing an intra-class mode dropping behavior. Li et al. [2019d] also note that the IS overfits within the context of text-to-image synthesis and can be "gamed" by increasing the batch size at the end of the training. Furthermore, the IS uses the output of the classification layer of an Inception-Net pre-trained on the ImageNet data set. This might not be the best approach for a more complex data set in which each image contains multiple objects at distinct locations throughout the image, as opposed to the ImageNet data set which consists of images usually depicting one object in the image center. Figure 4.7a shows some exemplary failure cases of the IS on images sampled from the COCO data set.

The FID relies on representative ground truth data to compare the generated data against and also assumes that features are of Gaussian distribution, which is often not the case. For more complex data sets the FID also still suffers from the problem that the image statistics are obtained with a network pre-trained on ImageNet which might not be a representative data set. Finally, neither the IS nor the FID take the image caption into account during their evaluation.

VS similarity and R-precision Zhang et al. [2018d] introduce the visual-semantic similarity (VS similarity) metric which measures the distance between a generated image and its caption. Two models are trained to embed images and captions respectively and then minimize the cosine distance between embeddings of matching image-caption pairs while maximizing the cosine distance between mismatching image-caption pairs. A good model then achieves high VS similarity between a generated image and its associated caption.

Xu et al. [2018b] use the R-precision metric to evaluate how well an image matches a given description or caption. The R-precision score is similar to VS similarity, but instead of scoring the VS similarity between a given image and caption it instead performs a ranking of the similarity between the real caption and randomly sampled captions for a given generated image. For this, first, an image is generated conditioned on a given caption. Then, another 99 captions are chosen



(a) Examples when IS fails for COCO images. The top row shows images for which the Inception-Net has very high entropy in its output layer, possibly because the images contain more than one object and are often not centered. The second row shows images containing different objects and scenes which were nonetheless all assigned to the same class by the Inception-Net, thereby negatively affecting the overall predicted diversity in the images.

(b) Examples when R-precision fails for COCO images. The top row shows images from the COCO data set. The middle row shows the correct caption and the bottom row gives examples for characteristics of captions that are rated as being more similar than the original caption.

Figure 4.7: Visualization of shortcomings of (a) the Inception Score (IS) and (b) the R-precision as evaluation metrics for text-to-image tasks.

randomly from the data set. Both the generated images and the 100 captions are then encoded with the respective image and text encoder. Similar to VS similarity the cosine distance between the image embedding and each caption embedding is used as proxy for the similarity between the given image and caption. The 100 captions are then ordered in descending similarity and the top k (usually k=1) most similar captions are used to calculate the R-precision. Intuitively, R-precision calculates if the real caption is more similar to the generated image (in feature space) than 99 randomly sampled captions.

The drawback of both metrics is that they do not evaluate the quality of individual objects. For example the real caption could state that *"a person stands on a snowy hill"* while the 99 random captions do not mention "snow" (which usually covers most of the background in the generated image) or "person" (but e.g. giraffe, car, bedroom, etc). In this case, an image with only white background (snow) would already make the real caption rank very highly in the R-precision metric and achieve a high VS similarity. See Figure 4.7b for a visualization of this. As such, this metric does not focus on the quality of individual objects but rather concentrates on global background and salient features.

Classification Accuracy Score Ravuri and Vinyals [2019] introduce the Classification Accuracy Score (CAS) to evaluate conditional image generation models, similar to Shmelkov et al. [2018]. For this, a classifier is trained on images generated by the conditional generative model. The classifier's performance is then evaluated on the original test set of the data set that was used to train the generative model. If the classifier achieves high accuracy on the test set, this indicates that the data it was trained on is representative of the real distribution. The authors

find that neither the IS, the FID, nor combinations thereof are predictive of the CAS, further indicating that the IS and FID are only of limited use for evaluating image quality.

Caption Generation Hong et al. [2018b] suggest evaluating text-to-image models by comparing original captions with captions obtained from generated images. The intuition is that if the generated image is relevant to its caption, then it should be possible to infer the original text from it. To this end, Hong et al. [2018b] use a pre-trained caption generator Vinyals et al. [2016] to generate captions for each synthesized image and compare these to the original ones through standard language similarity metrics, i.e. BLEU, METEOR, and CIDEr. Except for CIDEr, these metrics were originally developed to evaluate machine translation and text summarization methods and were only later adopted for the evaluation of image captions.

One challenge with this caption generation approach is that often many different captions are valid for a given image. Even if two captions are not similar, this does not necessarily imply that they do not describe the same image Anderson et al. [2016]. Furthermore, it has been shown that metrics such as BLEU, METEOR, and CIDEr are primarily sensitive to n-gram overlap which is neither necessary nor sufficient for two sentences to convey the same meaning [Giménez and Màrquez, 2007; Anderson et al., 2016; Madhyastha et al., 2019] and do also not necessarily correlate with human judgments of captions [Vinyals et al., 2016; Kilickaya et al., 2017]. Finally, there is no requirement that captions, either real or generated, need to focus on specific objects. Instead, captions can also describe the general layout of a given scene (e.g. *a busy street with lots of traffic*) without explicitly mentioning specific objects. Some of these limitations might potentially be overcome in the future by novel image caption evaluation metrics that focus more on objects and semantic content in the scene [Anderson et al., 2016; Madhyastha et al., 2019; Agarwal et al., 2020].

Other Approaches In contrast to the IS, which measures the diversity of a whole set of images, the diversity score [Zhao et al., 2019] measures the perceptual difference between a pair of images in feature space. This metric can be useful when images are generated from conditional inputs (e.g. labels or scene layouts) to examine whether a model can generate diverse outputs for a given condition. However, the metric does not convey anything directly about the quality of the generated images or their congruence with any conditional information. Chen and Koltun [2017]; Wang et al. [2018a], and Park et al. [2019] run a semantic segmentation network on generated images and compare the predicted segmentation mask to the ground truth segmentation mask used as input for the model. However, this metric needs a ground truth semantic segmentation mask and does not provide information about specific objects within the image.

Semantic Object Accuracy (SOA)

So far, most evaluation metrics are designed to evaluate the holistic image quality but do not evaluate individual areas or objects within an image. Furthermore,

except for *Caption Generation* and *R-precision*, none of the scores take the image caption into account when evaluating generated images. To address the challenges and issues mentioned above we introduce a novel evaluation metric based on a pre-trained object detection network. The pre-trained object detector evaluates images by checking if it recognizes objects that the image should contain based on the caption. For example, if the image caption is *"a person is eating a pizza"* we can infer that the image should contain both a person and a pizza and the object detector should be able to recognize both objects within the image. Since this evaluation measures directly whether objects specifically mentioned in the caption are recognizable in an image we call this metric *Semantic Object Accuracy* (SOA).

Some previous works have used similar approaches to evaluate the quality of the generated images. Hinz et al. [2019] evaluate how often expected objects (based on the caption) are detected by an object detector. However, only a subset of the captions is evaluated and the evaluated captions contain false positives (e.g. captions containing the phrase "hot dog" are evaluated based on the assumption that the image should contain a dog). Sah et al. [2018] introduce a detection score that calculates (roughly) whether a pre-trained object detector detects an object in a generated image with high certainty. However, no information from the caption is taken into account, meaning any detection with high confidence is "good" even if the detected object does not make sense in the context of the caption. Deng et al. [2018] use a pre-trained object detector to calculate the mean average precision and report precision-recall curves. However, the evaluation is done on synthetic data sets and without textual information as conditional input. Zhao et al. [2019] use classification accuracy as an evaluation metric in which they report the object classification accuracy in generated images. For this, they use a ResNet-101 model which is trained on real objects cropped and resized from the original data. However, in order to calculate the score, the size and location of each object in the generated image must be known, so this evaluation is not directly applicable to approaches that do not use scene layouts or similar representations. Vo and Sugimoto [2020] use recall and intersection-over-union (IoU) to evaluate the bounding boxes in their generated scene layout but do not apply these evaluations to generated images directly.

SOA Since we work with the COCO data set we filter all captions in the validation set for specific keywords that are related to the available labels for objects (e.g. person, car, zebra, etc). For each of the 80 available labels in the COCO data set we find all captions that imply the existence of the respective object and generate three images for each of the captions. The supplementary material gives a detailed overview of how exactly the captions were chosen for each label. We then run the YOLOv3 network [Redmon and Farhadi, 2018] pre-trained on the COCO data set on each of the generated images and check whether it recognizes the given object. We report the recall as a class average (SOA-C), i.e. in how many images per class the YOLOv3 on average detects the given object, and as an image average (SOA-I), i.e. on average in how many images a desired object was detected.

Specifically, the SOA-C is calculated as

$$\text{SOA-C} = \frac{1}{|C|} \sum_{c \in C} \frac{1}{|I_c|} \sum_{i_c \in I_c} \text{YOLOv3}(i_c), \tag{4.11}$$

for object classes $c \in C$ and images $i \in I_c$ that are supposed to contain an object of class c. The SOA-I is calculated as

$$\text{SOA-I} = \frac{1}{\sum_{c \in C} |I_c|} \sum_{c \in C} \sum_{i_c \in I_c} \text{YOLOv3}(i_c), \tag{4.12}$$

and

$$\text{YOLOv3}(i_c) = \begin{cases} 1 & \text{if YOLOv3 detected an object of class } c \\ 0 & \text{otherwise} \end{cases}. \tag{4.13}$$

Since many images can also contain objects that are not specifically mentioned (for example an image described by *"lots of cars are on the street"* could still contain persons, dogs, etc) in the caption we do not calculate a false negative rate but instead only focus on the recall, i.e. the true positives.

SOA-Intersection over Union Several approaches (e.g. Hinz et al. [2019]; Li et al. [2019d]; Hong et al. [2018b]; Zhao et al. [2019]; Vo and Sugimoto [2020]) use additional conditioning information such as scene layouts or bounding boxes. For these approaches, our evaluation metric can also calculate the intersection over union (IoU) between the location at which different objects should be and locations at which they are detected, which we call SOA-IoU. To calculate the IoU we use every image in which the YOLOv3 network detected the respective object. Since many images contain multiple instances of a given object we calculate the IoU between each predicted bounding box for the given object and each ground truth bounding box. The final IoU for a given image and object is then the maximum of the values, i.e. the reported IoU is an upper bound on the actual IoU.

Overall this approach allows a more fine-grained evaluation of the image content since we can now focus on individual objects and their features. To get a better idea of the overall performance of a model we calculate both the class average recall/IoU (SOA-C/SOA-IoU-C) and image average recall/IoU (SOA-I/SOA-IoU-I). Additionally, we report the SOA-C for the forty most and least common labels (SOA-C-Top40 and SOA-C-Bot40) to see how well the model can generate objects of common and less common classes.

4.3.5 Experiments

We perform multiple experiments and ablation studies. In a first step, we add the object pathway (OP) on multiple layers of the generator and to each discriminator and call this model *OPv2*. We also train this model with the additional bounding box loss we introduced in Subection 4.3.3. When the model is trained with the additional bounding box loss we refer to it as *BBL*.

Table 4.2: Inception Score (IS), Fréchet Inception Distance (FID), and R-precision on the MS-COCO data set. Results of our models are obtained with generated bounding boxes. Scores for models marked with † were calculated with a pre-trained model provided by the respective authors.

Model	IS ↑	FID ↓	R-precision (k=1) ↑
Original Images	34.88 ± 0.01	6.09 ± 0.05	68.58 ± 0.08
AttnGAN [Xu et al., 2018b]†	23.61 ± 0.21	33.10 ± 0.11	83.80
[Huang et al., 2019a]	23.74 ± 0.36		86.44 ± 3.38
ControlGAN [Li et al., 2019a]	24.06 ± 0.60		82.43
AttnGAN + OP [Hinz et al., 2019]†	24.76 ± 0.43	33.35 ± 1.15	82.44
MirrorGAN [Qiao et al., 2019b]	26.47 ± 0.41		74.52
Obj-GAN [Li et al., 2019d]†	24.09 ± 0.28	36.52 ± 0.13	87.84 ± 0.08
HfGAN [Huang et al., 2019b]	27.53 ± 0.25		
DM-GAN [Zhu et al., 2019]†	32.32 ± 0.23	27.34 ± 0.11	$\mathbf{91.87 \pm 0.28}$
SD-GAN [Yin et al., 2019]	$\mathbf{35.69 \pm 0.50}$		
OP-GAN (Best Model)	27.88 ± 0.12	$\mathbf{24.70 \pm 0.09}$	89.01 ± 0.26
OPv2, 0 obj	26.80 ± 1.01	30.01 ± 1.81	83.87 ± 1.22
OPv2, 1 obj	27.68 ± 0.47	26.18 ± 0.27	87.37 ± 0.60
OPv2, 3 obj	27.78 ± 0.50	26.45 ± 0.40	87.74 ± 1.08
OPv2, 10 obj	27.66 ± 0.34	26.52 ± 0.44	87.73 ± 0.98
OPv2 + *BBL*, 0 obj	24.60 ± 1.25	33.03 ± 0.76	81.27 ± 1.45
OPv2 + *BBL*, 1 obj	26.34 ± 0.55	26.59 ± 1.04	86.42 ± 0.60
OPv2 + *BBL*, 3 obj	26.52 ± 0.47	26.74 ± 1.08	87.08 ± 0.60
OPv2 + *BBL*, 10 obj	26.48 ± 0.58	26.83 ± 1.10	86.80 ± 0.56
OPv2 + *MO*, 0 obj	24.32 ± 1.65	35.36 ± 1.95	79.75 ± 1.87
OPv2 + *MO*, 1 obj	27.36 ± 0.49	25.06 ± 1.11	88.33 ± 0.81
OPv2 + *MO*, 3 obj	27.65 ± 0.37	24.96 ± 1.12	89.13 ± 0.42
OPv2 + *MO*, 10 obj	27.59 ± 0.43	$\mathbf{24.94 \pm 1.09}$	$\mathbf{89.14 \pm 0.41}$
OPv2 + *BBL* + *MO*, 0 obj	21.84 ± 0.83	45.79 ± 1.16	72.71 ± 1.75
OPv2 + *BBL* + *MO*, 1 obj	27.61 ± 0.67	26.19 ± 0.82	87.85 ± 0.25
OPv2 + *BBL* + *MO*, 3 obj	$\mathbf{28.04 \pm 0.65}$	25.91 ± 1.03	88.90 ± 0.24
OPv2 + *BBL* + *MO*, 10 obj	27.90 ± 0.79	25.80 ± 1.01	89.00 ± 0.17

Different approaches differ in how many objects per image are used during training. If an image layout is used, typically all objects (foreground and background) are used as conditioning information. Other approaches limit the number of objects during per training [Johnson et al., 2018; Hinz et al., 2019]. To examine the effect of training with different numbers of objects per image we train our approach with either a maximum of three objects per image (standard) or with up to ten objects per image, which we refer to as many objects (*MO*). When training with a maximum of three objects per image we sample randomly from the training set at train time, i.e. each batch contains images which contain zero to three objects.

Table 4.3: Caption Generation with CIDEr and Semantic Object Accuracy on Class (SOA-C) and Image Average (SOA-I) on the MS-COCO data set. Results of our models are obtained with generated bounding boxes. Scores for models marked with † were calculated with a pre-trained model provided by the respective authors.

Model	CIDEr ↑	SOA-C ↑	SOA-I ↑
Original Images	0.795 ± 0.003	74.97	80.84
AttnGAN [Xu et al., 2018b]†	0.695 ± 0.005	25.88	39.01
AttnGAN + OP [Hinz et al., 2019]†	0.689 ± 0.008	25.46	40.48
Obj-GAN [Li et al., 2019d]†	0.783 ± 0.002	27.14	41.24
DM-GAN [Zhu et al., 2019]†	$\mathbf{0.823 \pm 0.002}$	33.44	48.03
OP-GAN (Best Model)	0.819 ± 0.004	**35.85**	**50.47**
OPv2, 0 obj	0.760 ± 0.004	26.04 ± 1.47	37.56 ± 1.27
OPv2, 1 obj	0.798 ± 0.013		
OPv2, 3 obj	0.805 ± 0.011		
OPv2, 10 obj	0.806 ± 0.006	33.82 ± 0.69	48.39 ± 1.01
OPv2 + *BBL*, 0 obj	0.735 ± 0.029	24.00 ± 2.13	34.01 ± 2.89
OPv2 + *BBL*, 1 obj	0.783 ± 0.006		
OPv2 + *BBL*, 3 obj	0.793 ± 0.013		
OPv2 + *BBL*, 10 obj	0.794 ± 0.015	33.19 ± 0.40	48.24 ± 0.68
OPv2 + *MO*, 0 obj	0.695 ± 0.015	21.15 ± 1.47	30.24 ± 2.36
OPv2 + *MO*, 1 obj	0.789 ± 0.008		
OPv2 + *MO*, 3 obj	0.807 ± 0.014		
OPv2 + *MO*, 10 obj	0.805 ± 0.013	33.46 ± 1.01	47.93 ± 1.56
OPv2 + *BBL* + *MO*, 0 obj	0.626 ± 0.025	16.55 ± 1.81	22.76 ± 2.17
OPv2 + *BBL* + *MO*, 1 obj	0.791 ± 0.009		
OPv2 + *BBL* + *MO*, 3 obj	0.810 ± 0.009		
OPv2 + *BBL* + *MO*, 10 obj	$\mathbf{0.814 \pm 0.007}$	$\mathbf{34.51 \pm 1.12}$	$\mathbf{48.90 \pm 0.72}$

If an image contains more than three objects we choose the three largest ones in terms of area of the bounding box. When training with up to ten objects per image we slightly change our sampling strategy so that each batch consists of images that contain the same amount of objects. This means that, e.g., each image in a batch contains exactly four objects, while in the next batch each image might contain exactly seven objects. This increases the training efficiency as most of the images contain fewer than five objects.

As a result of the different settings we perform the following experiments:

1. *OPv2*: apply the object pathway (OP) on multiple layers of the generator and on all discriminators, training without the bounding box loss and with a maximum of three objects per image.

2. *OPv2* + *BBL*: same as *OPv2* but with the bounding box loss added to the discriminator loss term.

3. *OPv2 + MO*: same as *OPv2* but with a maximum of ten objects per image.

4. *OPv2 + BBL + MO* (*OP-GAN*): combination of all three approaches.

We train each model three times on the 2014 split of the COCO data set. At test time we use bounding boxes generated by a network [Li et al., 2019d] as the conditioning information. Therefore, except for the image caption no other ground truth information is used at test time.

4.3.6 Evaluation and Analysis

Table 4.2, Table 4.3, Table 4.4, and Table 4.5 give an overview of our results for the COCO data set. The first half of the table shows the results on the original images from the data set and from related literature while the second half shows our results. To make a direct comparison we calculated the IS, FID, CIDEr, and R-precision scores ourselves for all models which are provided by the authors. As such, the values from AttnGAN [Xu et al., 2018b], AttnGAN+OP [Hinz et al., 2019], Obj-GAN [Li et al., 2019d], and DM-GAN [Zhu et al., 2019] are the ones most directly comparable to our reported values since they were calculated in the same way.

Note that there is some inconsistency in how the FID is calculated in prior works. Some approaches, e.g. Li et al. [2019d], compare the statistics of the generated images only with the statistics of the respective "original" images (i.e. images corresponding to the captions that were used to generate a given image). We, on the other hand, generate 30,000 images from 30,000 randomly sampled captions and compare their statistics with the statistics of the full validation set. Many of the recent publications also do not report the FID or R-precision. This makes a direct comparison difficult as we show that the IS is likely the least meaningful score of the three since it easily overfits [Li et al., 2019d] and due to the reasons mentioned in Subection 4.3.4. We calculate each of the reported values of our models three times for each trained model (nine times in total) and report the average and standard deviation. To calculate the SOA scores we generate three images for each caption in the given class, except for the "person" class, for which we randomly sample 30,000 captions (from over 60,000) and generate one image for each of the 30,000 captions.

Quantitative Results

Overall Results As Table 4.2 and Table 4.3 show, all our models outperform the baseline AttnGAN in all metrics. The IS is improved by $16-19\%$, the R-precision by $6-7\%$, the SOA-C by $28-33\%$, the SOA-I by $22-25\%$, the FID by $20-25\%$, and CIDEr by $15-18\%$. This was achieved by adding our object pathways to the baseline model without any further tuning of the architecture, hyperparameters, or the training procedure. Our approach also outperforms all other approaches based on FID, SOA-C, and SOA-I. While there are two approaches that report a IS higher than our models, it has previously been observed that this score is likely

Table 4.4: Comparison of the recall values for the different models. We used generated bounding boxes to calculate the values. Numbers in brackets show scores when the object pathway was not used at test time.

Model	SOA-C / IoU	SOA-I / IoU
Original Images	74.97 / 0.550	80.84 / 0.570
AttnGAN [Xu et al., 2018b]	25.88 / ––	39.01 / ––
AttnGAN + OP [Hinz et al., 2019]	25.46 / 0.236	40.48 / 0.311
Obj-GAN [Li et al., 2019d]	27.14 / 0.513	41.24 / 0.598
DM-GAN [Zhu et al., 2019]	33.44 / ––	48.03 / ––
OPv2	33.82 (26.04) / 0.207	48.39 (37.56) / 0.270
OPv2 + BBL	33.19 (24.00) / 0.210	48.24 (34.01) / 0.270
OPv2 + MO	33.46 (21.15) / 0.214	47.93 (30.24) / 0.275
OPv2 + BBL + MO	34.51 (16.55) / 0.217	48.90 (22.76) / 0.278

Table 4.5: Comparison of the recall values for the different models. We used generated bounding boxes to calculate the values. Numbers in brackets show scores when the object pathway was not used at test time.

Model	SOA-C-Top40 / IoU	SOA-C-Bot40 / IoU
Original Images	78.77 / 0.546	71.18 / 0.554
AttnGAN [Xu et al., 2018b]	37.47 / ––	14.29 / ––
AttnGAN + OP [Hinz et al., 2019]	39.77 / 0.308	11.15 / 0.164
Obj-GAN [Li et al., 2019d]	39.88 / 0.587	14.40 / 0.438
DM-GAN [Zhu et al., 2019]	47.73 / ––	19.15 / ––
OPv2	48.34 (36.53) / 0.260	19.31 (15.55) / 0.152
OPv2 + BBL	47.96 (32.96) / 0.261	18.43 (15.04) / 0.159
OPv2 + MO	47.84 (28.15) / 0.264	19.07 (14.15) / 0.163
OPv2 + BBL + MO	49.70 (22.19) / 0.269	19.32 (10.91) / 0.165

the least meaningful for this task and can be gamed to achieve higher numbers [Barratt and Sharma, 2018; Li et al., 2019d]. Our user study also shows that the IS is the score that has the least predictive value for human evaluation.

We also calculated each score using the original images of the COCO data set. For the IS we sampled three times 30,000 images from the validation set and resized them to 256×256 pixels. These images were also used to calculate the CIDEr score. To calculate the FID we randomly sampled three times 30,000 images from the training set and compared them to the statistics of the validation set. The R-precision was calculated on three times 30,000 randomly sampled images and the corresponding caption from the validation set and the SOA-C and SOA-I were calculated on the real images corresponding to the originally chosen captions.

As we can see, the IS is close to the current state of the art models with a value

Figure 4.8: Comparison of images generated by different variations of our models.

of 34.88. It is possible to achieve a much higher IS on other, simpler data sets, e.g. IS > 100 on the ImageNet data set [Brock et al., 2019]. This indicates that the IS is indeed not a good evaluation metric, especially for complex images consisting of multiple objects and various locations. The difference between the R-precision on real and generated images is even larger. On the original images, the R-precision score is only 68.58, which is much worse than what current models can achieve (> 88).

One reason for this might be that the R-precision calculates the cosine similarity between an image embedding and a caption embedding and measures how often the caption that was used to generate an image is more similar than 99 other, randomly sampled captions. However, the same encoders that are used to calculate the R-precision are also used during training to minimize the cosine similarity between an image and the caption it was generated from. As a result, the model might already overfit to this metric through the training procedure. Our observation is that the models tend to heavily focus on the background to make it match a specific word in the caption (e.g. images tend to be very white when the caption mentions "snow" or "ski", very blue when the caption mentions "surf" or "beach", very green when the caption mentions "grass" or "savanna", etc.) This matching might lead to a high R-precision score since it leads, on average, to a large cosine similarity. Real images do not always reflect this, since a large part of the image might be occupied by a person or an animal, essentially "blocking out" the background information. We see a similar trend for the CIDEr evaluation where many models achieve a score similar to the score reached by real images. Regardless of what the actual reason is, the question remains whether evaluation metrics like the IS, R-precision, and CIDEr are meaning- and helpful when models that can not (as of now) generate images that would be confused as "real" achieve scores comparable to or better than real images.

The FID and the SOA values are the only two evaluation metrics (that we used) for which none of the current state of the art models can come close to the values obtained with the original images. The FID is still much smaller on the real data (6.09) compared to what current models can achieve (> 24 for the best models). While the FID still uses a network pre-trained on ImageNet it compares activations of convolutional layers for different images and is, therefore, likely still more meaningful and less dependent on specific object settings than the IS. Similarly, the SOA-C (SOA-I) on real data is 74.97 (80.84), while current models achieve values of around $30 - 36$ ($40 - 50$). Since the network used to calculate the SOA values is not part of the training loop the models can not easily overfit to this evaluation metric like they can for the R-precision. Furthermore, the results of the SOA evaluation confirm the impression that none of the models is able to generate images with multiple distinct objects of a quality similar to real images.

Impact of the Object Pathway To get a clearer understanding of how the evaluation metrics might be impacted by the object pathway we calculate our scores for a different number of generated objects. More specifically, we only apply the object pathway for a maximum given number of objects (0, 1, 3, or 10) per image. Intuitively, we would assume that without the application of the object pathway the IS and FID should be decreased, since the object pathway is not used to generate any object features and the images should, therefore, consist mostly of background. Additionally, we can get an intuition of how important the object pathway is for the overall performance of the network by looking at how it affects the R-precision, SOA-C, and SOA-I.

As Table 4.2 and Table 4.3 show, all models perform markedly worse when the object pathway is not used (0 obj). We find that the models trained with up to ten objects per image seem to rely more heavily on the object pathway than models trained with three objects per image. For models trained with only three objects per image (*OPv2* and *OPv2 + BBL*) the IS decreases by around $1 - 2$, the R-precision decreases by around $4 - 5$, the SOA-C (SOA-I) decreases by around $7 - 9$ ($11 - 14$), CIDEr decreases by around $6 - 8\%$, and the FID increases by around $4 - 7$. On the other hand, models trained with up to 10 objects suffer much more when the object pathway is removed, with the IS decreasing by around $3 - 6$, the R-precision decreasing by around $9 - 15$, the SOA-C (SOA-I) decreasing by around $12 - 18$ ($17 - 28$), CIDEr decreasing by around $16 - 30\%$, and the FID increasing by around $10 - 20$. These results indicate that the object pathways are an important part of the model and are responsible for at least some of the improvements compared to the baseline architecture.

Impact of Bounding Box Loss Adding the bounding box loss to the object pathways has a small negative effect on all scores, but does slightly improve the IoU scores (see Table 4.4 and Table 4.5). Note that the weighting of the bounding box loss in the overall loss term was not optimized but simply weighted with the same strength as the matching aware discriminator loss $\mathcal{L}_D^{\text{cls}}$. It is possible that the positive effect of the bounding box loss could be increased by weighting it differently.

Impact of Training on Many Objects Training the model with up to ten

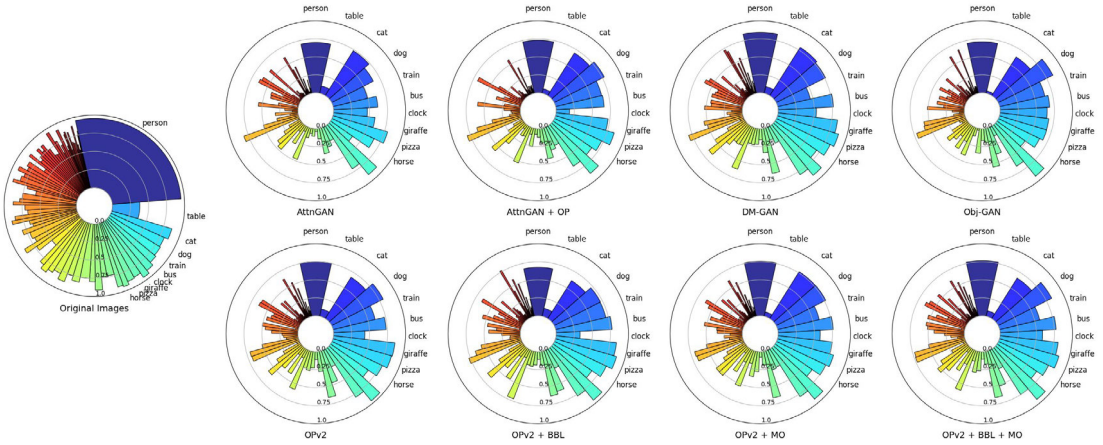


Figure 4.9: Comparison of SOA scores: SOA per class with degree of a bin reflecting relative frequency of that class.

Figure 4.10: Generated images and objects recognized by the pre-trained object detector (YOLOv3) which was used to calculate the SOA scores. The results highlight that, like most other CNN based object detectors, YOLOv3 focuses much more on texture and less on actual shapes.

objects per image has only minor effects on the IS and SOA scores, but improves the FID and R-precision. However, we observe that the models trained with only three objects per image slightly decrease in their performance once the object pathway is applied multiple times. Usually, the models trained on only three objects achieve their best performance when applying the object pathway three times as at training time. Once the model is trained on up to ten objects though, we do not observe this behavior anymore and instead achieve comparable or even better results when applying the object pathway ten times per image.

SOA Scores Table 4.4 shows the results for the SOA and SOA-IoU. The SOA-I values are consistently higher than the SOA-C values. Since the SOA-I is calculated on image average (instead of class average like the SOA-C) it is skewed by objects that often occur in captions and images (e.g. persons, cats, dogs, etc.). The SOA values for the most and least common 40 objects (Table 4.5) show that the models perform much better on the more common objects. Actually, most models perform about two times better on the common objects showing their problem in generating objects that are not often observed during training. For a detailed overview of how each model performed on the individual labels please refer to the supplementary material.

When we look at the IoU scores we see that the Obj-GAN [Li et al., 2019d] achieves by far the best IoU scores (around 0.5), albeit at the cost of lower SOA

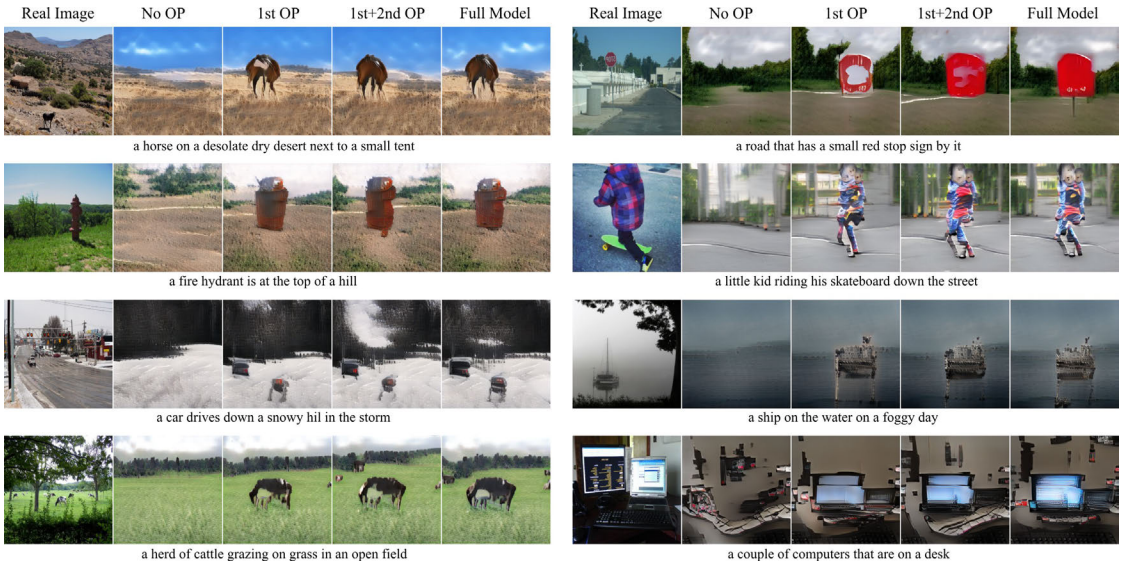


Figure 4.11: Comparison of images generated by our model (*OP-GAN*) with OPs switched on and off.

scores. Our models usually achieve an IoU of around $0.2 - 0.3$ on average. Training with up to ten objects per image and using the bounding box loss slightly increases the IoU. However, similar to previous work [Hinz et al., 2019; Li et al., 2019d] we find that the AttnGAN architecture tends to place salient object features at many locations of the image which affects the IoU scores negatively.

When looking at the SOA for individual objects (see Figure 4.9) we find that there are objects for which we can achieve very high SOA values (e.g. person, cat, dog, zebra, pizza, etc.). Interestingly, we find that all tested methods perform "good" or "bad" at the same objects. For example, all models perform reasonably well on objects such as *person* and *pizza* (many examples in the training set) as well as e.g. *plane* and *traffic light* (few examples in the training set). Conversely, all models fail on objects such as *table* and *skateboard* (many examples in the training set) as well as e.g. *hair drier* and *toaster* (few examples in the training set).

We found that objects need to have three characteristics to achieve a high SOA and the highest SOA scores are achieved when objects possess all three characteristics. The first important characteristic is easily predictable: the higher the occurrence of an object in the training data, the better (on average) the final performance on this object. Secondly, large objects, i.e. objects that usually cover a large part of the image (e.g. *bus* or *elephant*), are usually modeled better than objects that are usually small (*spoon* or *baseball glove*). The final and more subtle characteristic is the surface texture of an object. Objects with highly distinct surface textures (e.g. *zebra*, *giraffe*, *pizza*, etc.) achieve high SOA scores because the object detection network relies on these textures to detect objects. However, while the models are able to correctly match the surface texture (e.g. black and white stripes for a zebra) they are still not capable of generating a realistic-looking shape of many objects. As a result, many of these objects possess the "correct" surface texture but their shape is more a general "blob" consisting of the texture

and not a distinct form (e.g. a snout and for legs for a zebra). See Figure 4.10 for a visualization of this.

This is one of the weaknesses of the SOA score as it might give the wrong impression that an 80% object detection rate means in 80% of the cases the object is recognizable and of real-world quality. This is not the case, as the SOA scores are calculated with a pre-trained object detector which might focus more on texture and less on shapes of objects [Geirhos et al., 2019]. Consequently, the results of the SOA are more aptly interpreted as cases where a model was able to generate features that an independently pre-trained object detector would classify as a given object. The overall quality of the metric is, therefore, strongly dependent on the object detector and future improvements in this area might also lead to more meaningful interpretations of the SOA scores.

Figure 4.8 shows images generated by our different models. All images shown in this paper were generated without ground truth bounding boxes but instead use generated bounding boxes [Li et al., 2019d]. The first column shows the respective image from the data set, while the next four columns show the generated images. We can see that all models are capable of generating recognizable foreground objects. It is often difficult to find qualitative differences in the images generated by the different models. However, we find that the models using the bounding box loss usually improve the generation of rare objects. Training with ten objects per image usually leads to a slightly better image quality overall, especially for images that contain many objects.

As we saw in the quantitative evaluation, the object pathway can have a large impact on the image quality. Figure 4.11 shows what happens when (some of) the object pathways are not used in the full model ($OPv2 + BBL + MO$). Again, the first column shows the original image from the data set and the second column shows images generated without the use any of the object pathways. The next three columns show generated images when we consecutively use the object pathways, starting with the lowest object pathway and iteratively adding the next object pathway until we reach the full model. When no object pathway is used (first column) we clearly see that only background information is generated. Once the first object pathway is added we also get foreground objects and their quality gets slightly better by adding the higher-level object pathways.

User Study In order to further validate our results, we performed a user study on Amazon Mechanical Turk. Similar to other approaches [Zhang et al., 2018a; Yin et al., 2019; Hong et al., 2018b] we sampled 5,000 random captions from the COCO validation set. For each caption, we generated one image with each of the following models: our OP-GAN, the AttnGAN [Xu et al., 2018b], the AttnGAN-OP [Hinz et al., 2019], the Obj-GAN [Li et al., 2019d], and the DM-GAN [Zhu et al., 2019]. We showed each user a given caption and the respective five images (without time limit) from the models in random order and asked them to choose the image that depicts the given caption best. We evaluated each image caption twice, for a total of 10,000 evaluations with the help of 200 participants.

Table 4.6 shows how often each model was chosen as having produced the best image given a caption (variance was estimated by bootstrap [Efron, 1992]). This

Table 4.6: Human evaluation results (ratio of 1st by human ranking) of five models on the MS-COCO data set given a caption.

AttnGAN-OP [Hinz et al., 2019]	$14.65\% \pm 0.35$
AttnGAN [Xu et al., 2018b]	$16.80\% \pm 0.43$
Obj-GAN [Li et al., 2019d]	$20.96\% \pm 0.33$
DM-GAN [Zhu et al., 2019]	$22.42\% \pm 0.41$
OP-GAN (ours)	$\mathbf{25.17\% \pm 0.43}$

evaluation reveals that the human ranking closely reflects the ranking obtained through the SOA and FID scores. One notable exception are the two worst performing models (AttnGAN and AttnGAN-OP), which we measure to perform similar according to the SOA and FID scores, but obtain different results in the user study. We find that the IS score is not predictive of the performance in the user study. The R-precision and CIDEr are somewhat predictive, but predict a different ranking of the top-three performing models. Overall, we find that our OP-GAN performs best according to both the SOA scores and the human evaluation. As hypothesized in Subection 4.3.4 we also observe that the FID and SOA scores are the best predictors for a model's performance in a human user evaluation.

Qualitative Results

Figure 4.12 shows examples of images generated by our model (*OPv2* + *BBL* + *MO*) and those generated by several other models [Zhu et al., 2019; Li et al., 2019d; Hinz et al., 2019; Xu et al., 2018b]. We observe that our model often generates images with foreground objects that are more recognizable than the ones generated by other models. For more common objects (e.g. person, bus or plane) all models manage to generate features that resemble the object but in most cases do not generate a coherent representation from these features and instead distribute them throughout the image. As a result, we notice features that are associated with an object but not necessarily form one distinct and coherent appearance of that object. Our model, on the other hand, is often able to generate one (or multiple) coherent object(s) from the features, see e.g. the generated images containing a bus, cattle, or the plane.

When generating rare objects (e.g. cake or hot dog) we observe that our model generates a much more distinct object than the other models. Indeed, most models fail completely to generate rare objects and instead only generate colors associated with these objects. Finally, when we inspect more complex scenes we see that our model is also capable of generating multiple diverse objects within an image. As opposed to the other images for "*room showing a sink and some drawers*" we can recognize a sink-like shape and drawers in the image generated by our model. Similarly, our model can also generate an image containing a reasonable shape of a banana and a cup of coffee, whereas the other models only seem to generate the texture of a banana without the shape and completely ignore the cup of coffee.

Figure 4.12: Comparison of images generated by our model (*OP-GAN*) with images generated by other current models.

4.3.7 Conclusion

In this paper, we introduced a novel GAN architecture (*OP-GAN*) that specifically models individual objects based on some textual image description. This is achieved by adding object pathways to both the generator and discriminator which learn features for individual objects at different resolutions and scales. Our experiments show that this consistently improves the baseline architecture based on quantitative and qualitative evaluations.

We also introduce a novel evaluation metric named *Semantic Object Accuracy* (SOA) which evaluates how well a model can generate individual objects in images. This new SOA evaluation allows to evaluate text-to-image synthesis models in more detail and to detect failure and success modes for individual objects and object classes. A user study with 200 participants shows that the SOA score is consistent with the ranking obtained by human evaluation, whereas other scores such as the Inceptions Score are not. Evaluation of several state-of-the-art approaches using SOA shows that no current approach is able to generate realistic foreground objects for the 80 classes in the COCO data set. While some models achieve high accuracy for several of the most common objects, all of them fail when it comes to modeling rare objects or objects that do not have an easily recognizable surface structure. However, using the SOA as an evaluation metric on text-to-image models provides more detailed information about how well they perform for different object classes or image captions and is well aligned with human evaluation.

4.4 Intermediate Discussion

Section 4.2 shows that using compositional representations for GANs has advantages such as higher image quality, better generalization, and more control over the generation process. Section 4.3 focuses more on how we can improve the automatic

evaluation of object-based compositional representations in generative models by introducing the novel Semantic Object Accuracy (SOA) score. We now focus on several open challenges for compositional representations and generative models and how they might be addressed in the future.

Evaluation of Compositional Representations Previous evaluation metrics for compositional representations focused only on discriminative tasks such as image classification but were not able to evaluate images generated from compositional representations. We introduced a score – Semantic Object Accuracy (SOA) – that, for the first time, evaluates the alignment between generated images and textual descriptions based directly on objects mentioned specifically in the textual description [Hinz et al., 2020]. SOA allows for a more explicit evaluation of compositional representations as it explicitly evaluates the different compositional parts (in our case objects). Through this, we can determine which objects (i.e. which parts of the compositional representations) are modeled better or worse compared to other objects. This allows us to identify specific criteria (e.g. size and texture) that are helpful for better modeling individual objects.

However, SOA does not solve all challenges with evaluating image quality, image-text alignment, and compositional representations. Strictly speaking, SOA only evaluates whether a pre-trained object detector detects a specific object within a given image. Whether the detected object is of "good" quality depends on the used object detector (see e.g. Figure 4.10). SOA also does not evaluate the image background or whether interactions between objects are modeled correctly. Furthermore, SOA does not evaluate "false positives", i.e. it does not evaluate whether the image contains objects that were not specifically mentioned in the text caption. This was, to a degree, a conscious decision on our part, as many images can be realistic despite containing objects that were not explicitly mentioned in the caption. For example, an image generated from the caption "*This image shows a kitchen*" may very well contain a fridge and microwave, despite these objects not being explicitly mentioned in the caption. However, it is unlikely (though not impossible) that this image contains, e.g., a "zebra" or a "motorbike". As we found it difficult (if not impossible) to manually decide and label which object combinations in a given image are realistic or not we decided to focus only on real positives as a proxy of image-text alignment. Our user study shows that this still results in strong alignment with human perception while making it possible to easily scale the SOA to even more challenging datasets as e.g. done by Zhang et al. [2021] who apply the SOA to the challenging LN-COCO data set [Pont-Tuset et al., 2020].

Similar to disentangled representations, another challenge with evaluating compositional representations is the fact that it is not necessarily always clear what constitutes an "object". In our evaluation "objects" were defined by the available labels in the data set, but of course individual "objects" often consist of other "objects" (e.g. a car consists of tires, chassis, ...). We do, therefore, not evaluate compositionality in the part-based sense, but only in the object-based sense. How to evaluate compositional representations in this setting is likely dependent on the

task at hand, available labels, and the practitioner's implicit assumptions about what "objects" are in the given domain.

Overall, just as evaluating image generation in itself is already a challenging problem [Theis et al., 2016], evaluating compositional representations and image-text alignment are challenging in their own ways [Hinz et al., 2020; Frolov et al., 2021]. Current evaluation methods only address small areas of evaluation (e.g. only image quality, only image-text alignment, ...) and we need a combination of different metrics to evaluate different quality aspects. Other evaluation metrics, e.g. generalization to unseen compositional representations (e.g. combining objects in a scene that were not seen together at train time or placing objects at locations not seen at train time) are only evaluated visually [Hinz et al., 2019] or, most often, are not evaluated at all [Casanova et al., 2020]. The SOA is a first step to a more robust and humanly interpretable evaluation metric that is well aligned with human perception and allows for quality evaluation both on a global image perspective and for individual objects.

Required Labeling The number of labels needed to learn compositional representations can directly affect their popularity and long-term success. Fewer requirements for labels will likely increase the adoption of compositional representations while requiring many hand-annotated labels will likely reduce it. Our current approach requires bounding boxes for all objects which requires either manual annotation or a network that can predict the bounding boxes (likely also trained on manual bounding box labels). Ideally, we would like to reduce the number of needed labels. Currently, some approaches try to leverage attention mechanisms in order to learn object-based representations with only textual guidance [Lee et al., 2018; Li et al., 2019d] but these approaches typically do not learn compositional representations. Nguyen-Phuoc et al. [2020], Ehrhardt et al. [2020], and Van Steenkiste et al. [2020] learn object-based representation in a completely unsupervised way but only show results on synthetic or low-resolution data sets. A promising future direction would be to look into how we can apply generative models with the necessary inductive biases for object-based modeling to learn compositional representations of complex natural scenes without the need for manual labels.

Structure of Compositional Representations It is still an open question what the best structure for compositional representations in the GAN framework looks like and how to condition the generator and discriminator on them. In our approaches, we learn individual representations for each object and one additional representation for the background. In the generator, all these representations are then concatenated to generate the final output. The discriminator also gets a concatenation of all learned representations at the last layer to make its final decision. Overall, this might not be the best way, as this limits interactions between the different objects and the background since all of them are learned in isolation. This could be addressed by examining different representation merging approaches that do not rely on simple concatenation at a late stage of the model.

As mentioned in Subection 2.2.2, different GAN frameworks use different conditioning mechanisms. Our approaches use a very simple form of conditioning

that is given to the discriminator once at a late stage of the model. Exploring other mechanisms [Odena et al., 2017; Miyato and Koyama, 2018; Kang and Park, 2020] of incorporating the conditioning information, in our case both textual and object-specific, might help the discriminator to learn higher-order interactions between objects and, as a consequence, learn better representations. Furthermore, the generator also learns the low-level object representations in isolation, which might limit the learning capabilities. Investigating early-stage fusion techniques for representations in the generator might also lead to increased performance in this case.

4.5 Summary

Compositional representations are a powerful way of modeling complex objects or scenes. They allow for modularity, reusability, and more explicit control over how the world is represented. We show in Section 4.2 how they can improve the quality and generalization capabilities of modeling contexts that consist of individual objects. However, our model needs data where the individual objects are labeled with a class label and a bounding box, which limits its applicability in the real world. How our model learns compositional representations is also likely not ideal, as it learns the representations for individual objects in isolation. As such, future approaches should address these challenges by reducing the dependency on labeled data and by incorporating techniques that allow for interactions between representations at train time.

Section 4.3 illustrates the weaknesses of some of the current evaluation metrics and introduces a new metric that addresses several of the current shortcomings. Our metric can evaluate certain aspects better than other metrics, allows us to identify characteristics that make learning of compositional representations easier, and also aligns better with human evaluations. However, as noted in Section 4.4, there are still several weaknesses that need to be addressed to improve the quantitative evaluation of compositional representations.

Despite these challenges in learning and evaluating compositional representations, we believe that compositional representations are an important part of learning to represent the real world. They have several advantages compared to traditional representations learned by current deep neural networks (see Section 4.1) and are able to naturally model complex contexts. Future work should look into how we can use powerful deep neural networks to learn useful compositional representations without the need for large-scale human annotations. The inherently hierarchical learning approach taken by current deep neural networks lends itself to be extended to learn hierarchical compositional representations. This could potentially lead to a combination of part-based and object-based compositional representations where lower layers learn part-based representations which are combined to object-based representations on higher layers.

The potential usefulness of compositional representations is strongly dependent on the quality of the learned object representations. In this chapter, the object

representations were learned as part of a larger model that is trained on a large data set of complex scene layouts. However, since the learned representations are compositional we could, theoretically, also use already learned object representations in a plug-and-play manner. If the object representations are "good enough" this could improve the final model quality and would potentially make it possible to extend the model with data about objects that were not seen during training. This approach could also help with objects that are difficult to learn from a given data set (e.g. because they are too small or there are too few samples). In the next chapter (Chapter 5) we develop several approaches to learning object-specific representations, i.e. representations for a single object, from little data.

Chapter 5

Object-Specific Representations

In our previous approaches (Chapter 3 and Chapter 4) we trained our models on large data sets of images containing many different objects. However, many applications and domains do not need or provide large data sets, e.g. when we work with specific artistic drawings or want to learn a model for a single, specific object. Additionally, we showed in Section 4.3 that learning object representations can be challenging for some objects, e.g. when they have a complex shape or only take up a small part of the image. To address these challenges, it is beneficial to train a model to only learn a representation of the given context we are interested in. This can help reduce the amount of data that is needed and can yield a highly specialized model for the given context. Since the representation is learned for a specific object we can also process our training data specifically for this object (e.g. such that it takes up a reasonably large area of the image). We now describe different ways in which this problem can be approached, before presenting our two approaches and their position in the current literature in more detail.

5.1 Single-Object Representations

Traditional Generative Adversarial Networks (GANs) need thousands of training samples in order to learn a good representation that allows for sampling from the training distribution. However, in certain domains, e.g. drawings by a certain artist or medical image data, we do not have a lot of data available. Furthermore, for some tasks, e.g. editing a single image or harmonizing contents into a single image, we might not necessarily need a network that was trained on a large data set. In these cases it might be useful to train a model on only the relevant data, even if there is only very little data available. In the limit this could mean training a model on a single image or on several images of a single object. Being able to do this would allow more specialization of a given model, reduce the required amount of training data, and could help in making the resulting models smaller more parameter efficient.

Single-Object Representations In this chapter we use the term "single-object representation" to define the representation(s) that are learned by a model that was

trained on only a single object. We make no limitations of what exactly constitutes an "object" in this case: it could be a single object from a given domain (e.g. a chair or a face), but also a more complex "object" such as a landscape or a building. In the extreme case this could mean that the model is only trained on a single data point (Section 5.2). Alternatively, it could also mean that the model is trained on several different views of the given object, e.g. by changing the viewpoint or pose of the given object (Section 5.3). There are several different ways in how this challenge can be addressed, differing in the kind of data that we have available, the models and implicit biases that we choose, and the task that we want to solve.

Training Data The exact nature of the available training data strongly affects what the model can learn about the given object. The available training data can differ in how much information a single data point contains (e.g. 2D vs. 3D) and in how much training data we have available (only one view or multiple views). In the most restrictive setting we might only have a single 2D image without any additional information as training data [Shaham et al., 2019; Hinz et al., 2021b]. This can be extended by providing multiple views of the given object (e.g. from different viewpoints) to provide more knowledge about the object [Mildenhall et al., 2020; Poursaeed et al., 2020; Hinz et al., 2021a]. Alternatively the data that we get as input might already contain 3D information, e.g. by being represented as a point cloud or voxels [Sitzmann et al., 2019a]. Finally, we might have a time component in the data that provides us with even more information, e.g. how the object behaves throughout time [Li et al., 2020c]. In our approaches we work with 2D images in both a single-view (Section 5.2) and a multi-view (Section 5.3) setting.

Models and Implicit Biases Depending on the input data, the object we want to model, and the downstream task different models and implicit biases might be most useful. GANs can be trained to learn good representations from a single or few data points [Shaham et al., 2019; Hinz et al., 2021a,b; Vinker et al., 2020; Benaim et al., 2020; Shocher et al., 2019]. However, they mostly rely on standard convolutional neural networks and often do not learn a 3D representation of the data. Current implicit models [Sitzmann et al., 2019b, 2020b; Mildenhall et al., 2020], on the other hand, have a strong implicit bias towards representing objects in a continuous 3D manner. However, it is challenging to scale them to larger and more complex objects or scenes and the sampling at test time can take a long time. Our approaches are based on GANs and make no assumption about the 3D structure of the learned object. We discuss potential approaches to address this, e.g. by combining traditional GANs and implicit models, in the discussion (Section 5.4).

5.2 Learning Representations from a Single Data Point

This section presents our work *Improved techniques for training single-image GANs* by Tobias Hinz, Matthew Fisher, Oliver Wang, and Stefan Wermter published in

2021 at the *IEEE Winter Conference on Applications of Computer Vision (pp. 1300 – 1309)*.

5.2.1 Introduction

Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] are capable of generating realistic images [Brock et al., 2019] that are often indistinguishable from real ones [Karras et al., 2020b]. The resulting models can be used for different tasks, such as unconditional and conditional image synthesis [Karras et al., 2019; Hinz et al., 2019], image inpainting [Demir and Unal, 2018], and image-to-image translation [Isola et al., 2017; Zhu et al., 2017a]. However, most GANs are trained on large datasets, typically consisting of tens of thousands of images which can be time-consuming and expensive. In some cases, it might be preferable to train a generative model on a small number of images or, in the limit, on a single image. This is useful if we want to obtain variations of a given image, work with a very specific image or style, or only have access to little training data. The recently proposed SinGAN [Shaham et al., 2019] introduces a GAN that is trained on a single image for tasks such as unconditional image generation and harmonization.

SinGAN is trained in a multi-stage and multi-resolution approach, where the training starts at a very low resolution (e.g. 25×25 pixels) at the first stage. The training progresses through several "stages", at each of which more layers are added to the generator and the image resolution is increased. At each stage all previously trained stages (i.e. the generator's lower layers) are frozen and only the newly added layers are trained. We find that exactly how multi-stage and multi-resolution training is handled is critical. In particular, training only one stage at a given time limits interactions between different stages, and propagating images instead of feature maps from one generator stage to the next negatively affects the learning process. Conversely, training all stages end-to-end causes overfitting in the single image scenario, where the network collapses to generating only the input image. We experiment with this balance, and find a promising compromise, training multiple stages in parallel with decreased learning rates, and find that this improves the learning process, leading to more realistic images with less training time. Furthermore, we show how it is possible to directly trade-off image quality for image variance, where training more stages in parallel means a higher global image consistency at the price of less variation.

We also conduct experiments over the choice of rescaling parameters, i.e. how we decide at which image resolution to train at each stage. We observe that the quality of the generated images, especially the overall image layout, quickly degrades when there are not enough training stages with small resolution. Our experiments show that lower stages with smaller resolutions are important for the overall image layout, while higher stages with larger resolution are important for the final image texture and color. We find that we only need relatively few training stages with high-resolution images in order to still generate images with the correct texture. As a consequence, we put a higher weight on smaller resolution images during training while using fewer of the stages to train on high-resolution images.

Finally, since our model trains several stages in parallel, we can introduce a *task-specific fine-tuning stage* which can be performed on any trained model. For several tasks we show how to fine-tune the trained model on a given specific image to further improve results. This shows benefits with as few as 500 additional training iterations and is, therefore, very fast (less than two minutes on our hardware).

Combining these proposed architecture and training modifications enables us to generate realistic images with fewer stages and significantly reduced overall training time (20-25 minutes versus 120-150 minutes in the original SinGAN work). To summarize, our main contributions are:

1. We train several stages in parallel with different learning rates and can trade-off the variance in generated images vs. their conformity to the original training image.

2. We do not generate images at intermediate stages but propagate features directly from one stage to the next.

3. We improve the rescaling approach for multi-stage training, which enables us to train on fewer stages.

4. We introduce a fine-tuning phase which can be used on pre-trained models to obtain optimal results for specific images and tasks.

5.2.2 Related Work

Learning the statistics and distribution of patches of a single image has been known to provide a powerful prior since the empirical entropy of patches inside a single image is smaller than the empirical entropy of patches inside a distribution of images [Zontak and Irani, 2011]. By using this prior, many tasks such as inpainting [Ulyanov et al., 2018; Zhang et al., 2019], denoising [Zontak et al., 2013], deblurring [Michaeli and Irani, 2014], retargeting [Mastan and Raman, 2019, 2020], and segmentation [Gandelsman et al., 2019] can be solved with only a single image. In particular, image super-resolution [Yang et al., 2019; Huang et al., 2015; Glasner et al., 2009; Shocher et al., 2018; Bell-Kligler et al., 2019] and editing [Cho et al., 2008; Dekel et al., 2015; He and Sun, 2012; Mechrez et al., 2019; Tlusty et al., 2018; Mao et al., 2019a] from a single image have been shown to be successful and a large body of work focuses specifically on this task. Recent work also shows that training a model on a single image with self-supervision and data augmentation can be enough to learn powerful feature extraction layers [Asano et al., 2020].

Approaches that train GAN models on single images are still relatively rare and are usually based on a bidirectional similarity measure for image summarization [Simakov et al., 2008]. Some approaches do not use natural images, but instead train only on texture images [Jetchev et al., 2016; Zhou et al., 2018; Bergmann et al., 2017; Li and Wand, 2016]. At this time, only few models are capable of being trained on a single 'natural' image [Shaham et al., 2019; Shocher et al., 2019; Vinker et al., 2020]. Other novel approaches target applications such as image-to-image

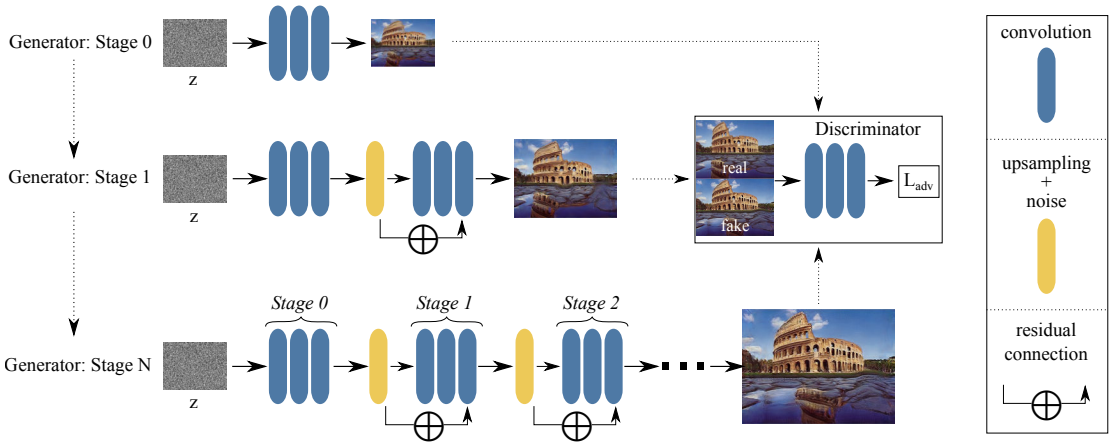


Figure 5.1: Overview of our model (ConSinGAN). We start training at 'Stage 0' with a small generator and small image resolution. With increasing number of stages both the generator capacity and image resolution increase.

translation with only two images as training data [Lin et al., 2020; Benaim et al., 2020].

The work most relevant to our approach is SinGAN [Shaham et al., 2019] which is the only model that can perform unconditional image generation after being trained on a single natural image. SinGAN trains both the generator and the discriminator over multiple stages of different image resolutions as it is useful to learn statistics of image patches across different image scales [Bagon et al., 2008]. The output at each stage is an image which is used as input to the next stage and each stage is trained individually while the previous stages are kept frozen.

5.2.3 Methodology

We now describe our findings in more detail, starting with the training of a multi-stage architecture, followed by best practices we found for scaling learning rate and image resolutions at different stages during training.

Multi-stage Training Multi-scale image generation is of critical importance [Shaham et al., 2019], however, there are many ways in which this can be realized. SinGAN only trains the current (highest) stage of its generator and freezes the parameters of all previous stages. ProGAN [Karras et al., 2018] presents a progressive growing scheme that adds levels with all weights unfrozen, and more recently [Karnewar and Wang, 2020; Karras et al., 2019] train the entire pyramid jointly.

In this work, we investigate whether the model can be trained end-to-end, rather than with training being fixed at intermediate stages, even in the single image task. However, we find that training all stages leads to *overfitting* (see Figure 5.3), i.e. the generator only generates the original training image without any variation. We develop a novel progressive growing technique that trains multiple, *but not all*, stages concurrently while simultaneously using progressively smaller learning rates

at lower stages. Since we train several stages of our model concurrently for a single image we refer to our model as 'Concurrent-Single-Image-GAN' (ConSinGAN).

Training ConSinGAN starts on a coarse resolution for a number of iterations, learning a mapping from a random noise vector z to a low-resolution image (see "Generator: Stage 0" in Figure 5.1). Once training of stage n has converged, we increase the size of our generator by adding three additional convolutional layers. In contrast to SinGAN, each stage gets the raw features from the previous stage as input, and previous layers are not fixed. We add a residual connection [He et al., 2016] from the original features to the output of the newly added convolutional layers (see "Generator: Stage 1" in Figure 5.1). We repeat this process N times until we reach our desired output resolution. We add additional noise to the features at each stage [Isola et al., 2017; Zhu et al., 2017b] to improve diversity. In our default setting, we jointly train the last three stages of a generator (see "Generator: Stage N" in Figure 5.1). While it is possible to train more than three stages concurrently, we observed that this rapidly leads to severe overfitting (Figure 5.3).

We use the same patch discriminator [Isola et al., 2017] architecture and loss function as the original SinGAN. This means that the receptive field in relation to the size of the generated image gets smaller as the number of stages increases, meaning that the discriminator focuses more on global layout at lower resolutions and more on texture at higher resolutions. In contrast to SinGAN we do not increase the capacity of the discriminator at higher stages, but use the same number of parameters at every stage. We initialize the discriminator for a given stage n with the weights of the discriminator of the previous stage $n-1$ at all stages. At a given stage n, we optimize the sum of an adversarial and a reconstruction loss:

$$\min_{G_n} \max_{D_n} \mathcal{L}_{adv}(G_n, D_n) + \alpha \mathcal{L}_{rec}(G_n). \tag{5.1}$$

$\mathcal{L}_{adv}(G_n, D_n)$ is the WGAN-GP adversarial loss [Gulrajani et al., 2017], while the reconstruction loss is used to improve training stability ($\alpha = 10$ for all our experiments). For the reconstruction loss the generator G_n gets as input a downsampled version (x_0) of the original image (x_N) and is trained to reconstruct the image at the given resolution of stage n:

$$\mathcal{L}_{rec}(G_n) = ||G_n(x_0) - x_n||_2^2. \tag{5.2}$$

The discriminator is always trained in the same way, i.e. it gets as input either a generated or a real image and is trained to maximise $\mathcal{L}_{adv}$. Our generator, however, is trained slightly differently depending on the final task.

Task Specific Generator Training For each task we use the original image x_n for the reconstruction loss $\mathcal{L}_{rec}$. The input for the adversarial loss $\mathcal{L}_{adv}$, however, depends on the task. For unconditional image generation the input to the generator is simply a randomly sampled noise vector for $\mathcal{L}_{adv}$. However, we found that if the desired task is known beforehand, better results can be achieved by training with a different input format. For example, for image harmonization, we can instead train using the original image with augmentation transformations applied as input. The

intuition for this is that a model that is used for image harmonization does not need to learn how to generate realistic images from random noise, but rather should learn how to harmonize different objects and color distributions. To simulate this task, we apply random combinations of augmentation techniques such as additive noise and color transforms to the original image x_N at each iteration. The generator gets the augmented image as input and needs to transform it back to an image that should resemble the original distribution.

Learning Rate Scaling The space of all learning rates for each stage is large and has a big impact on the final image quality. At any given stage n, we found that instead of training all stages (n, $n-1$, $n-2$, ...) with the same learning rate, using a lower learning rate on earlier stages ($n-1$, $n-2$, ...) helps reduce overfitting. If the learning rate at lower stages is too large (or too many stages are trained concurrently), the model generator quickly collapses and only generates the training image (Figure 5.3). Therefore, we propose to scale the learning rate η with a factor δ. This means that for generator G_n stage n is trained with learning rate $\delta^0\eta$, stage $n-1$ is trained with a learning rate $\delta^1\eta$, stage $n-2$ with $\delta^2\eta$, etc. In our experiments, we found that setting $\delta = 0.1$ gives a good trade-off between image fidelity and diversity (see Figure 5.3 and Figure 5.4).

Improved Image Rescaling Another critical design choice is around what kind of multiscale pyramid to use. SinGAN originally proposes to downsample the image x by a factor of r^{N-n} for each stage n where r is a scalar with default value 0.75. As a result, SinGAN is usually trained on eight to ten stages for a resolution of 250 width or height. When the images are downsampled more aggressively (e.g. $r = 0.5$) fewer stages are needed, but the generated images lose much of their global coherence.

We observe that this is the case when there are not enough stages at low resolution (roughly fewer than 60 pixels at the longer side). When training on images with a high resolution, the global layout is already "decided" and only texture information is important since the discriminator's receptive field is always 11×11. To achieve a certain global image layout we need a certain number of stages (usually at least three) at low resolution, but we do not need many stages a high resolution. We adapt the rescaling to not be strictly geometric (i.e. $x_n = x_0 \times r^{N-n}$), but instead to keep the density of low-resolution stages higher than the density of high-resolution stages:

$$x_n = x_N \times r^{((N-1)/log(N))*log(N-n)+1} \text{ for } n = 0, ..., N-1 \tag{5.3}$$

For example, with a rescaling scalar $r = 0.55$ we get six stages with the following resolutions and we observe that our new rescaling approach (second line) has more stages with smaller resolutions compared to the original rescaling approach (first line):

$$25\times34, 38\times50, 57\times75, 84\times112, 126\times167, 188\times250,$$
$$25\times34, 32\times42, 42\times56, 63\times84, \quad 126\times167, 188\times250.$$

Model	Confusion ↑	SIFID ↓	Train Time	# Stages	# Paramers
ConSinGAN	$16.0\% \pm 1.4\%$	0.06 ± 0.03	24 min	5.9	~660,000
SinGAN	$17.0\% \pm 1.5\%$	0.09 ± 0.07	152 min	9.7	~1,340,000

Table 5.1: Results of our user study and SIFID on images from the **Places** dataset.

To summarize our main findings, we produce feature maps rather than images at each stage, we train multiple stages concurrently, we propose a modified rescaling pyramid, and we present a task-specific training variation.

Figure 5.2: Example of unconditionally generated images showing complex global structure generated by ConSinGAN.

5.2.4 Results

We evaluate ConSinGAN on unconditional image generation and image harmonization in detail. For space reasons we focus on these two applications but note that other applications are also possible with ConSinGAN. We show examples of other tasks such as image retargeting, editing, and animation in the supplementary material.

Unconditional Image Generation

Since our architecture is completely convolutional we can change the size of the input noise vector to generate images of various resolutions at test time. Figure 5.2 shows an overview of results from our method on a set of challenging images that

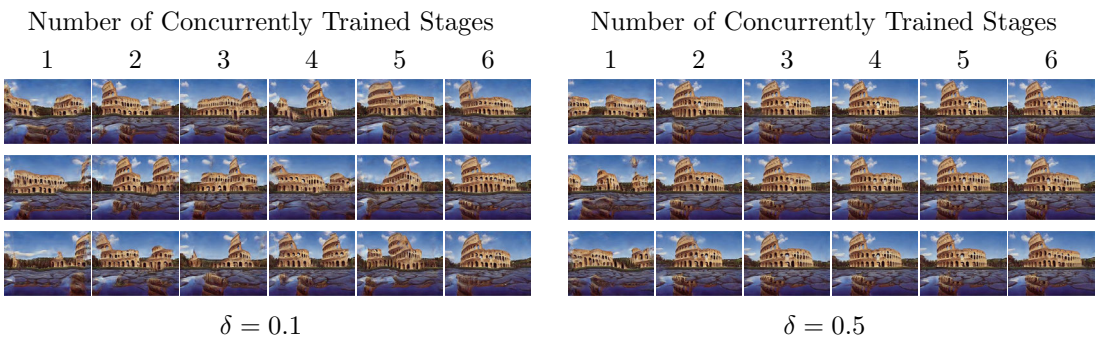


Figure 5.3: Effect of learning rate scale δ and concurrently trained stages for a model with six stages. Images are randomly selected.

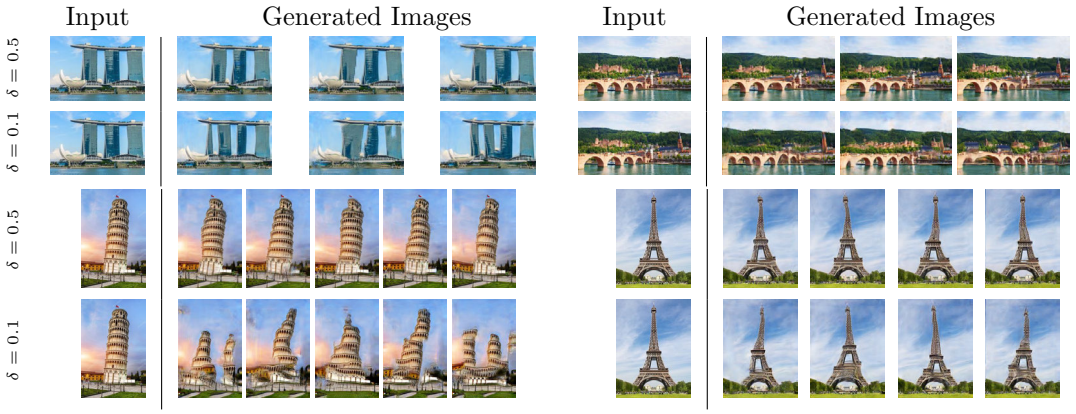


Figure 5.4: Effect of the learning rate scale δ during training of ConSinGAN.

require the generation of *global* structures for the images to seem realistic. We observe that ConSinGAN is successfully able to capture these global structures, even if we modify the image resolution at test time. For example, in the Stonehenge example, we can see how "stones" are added when the image width is increased and "layers" are added to the aqueduct image when the image height is increased.

Ablation We further examine the interplay between the learning rate scaling and the number of concurrently trained stages (Figure 5.3) and evaluate how varying the learning rate scaling δ (Subection 5.2.3) affects training (Figure 5.4). As we can see in Figure 5.3, training with a $\delta = 0.1$ leads to diverse images for most settings, with the diversity slightly decreasing with a larger number of concurrently trained stages. When training with $\delta = 0.5$ we observe a large decrease in image diversity even when only training two stages concurrently. As such, the number of concurrently trained stages and the learning rate scaling δ offer a trade-off between diversity and fidelity of the generated images.

Figure 5.4 visualizes how the variance in the generated images increases with decreasing δ for a model with three concurrently trained stages. For example, when we look at the top left example (Marina Bay Sands), we observe that for a $\delta = 0.5$ the overall layout of the image stays the same, with minor variations in, e.g., the

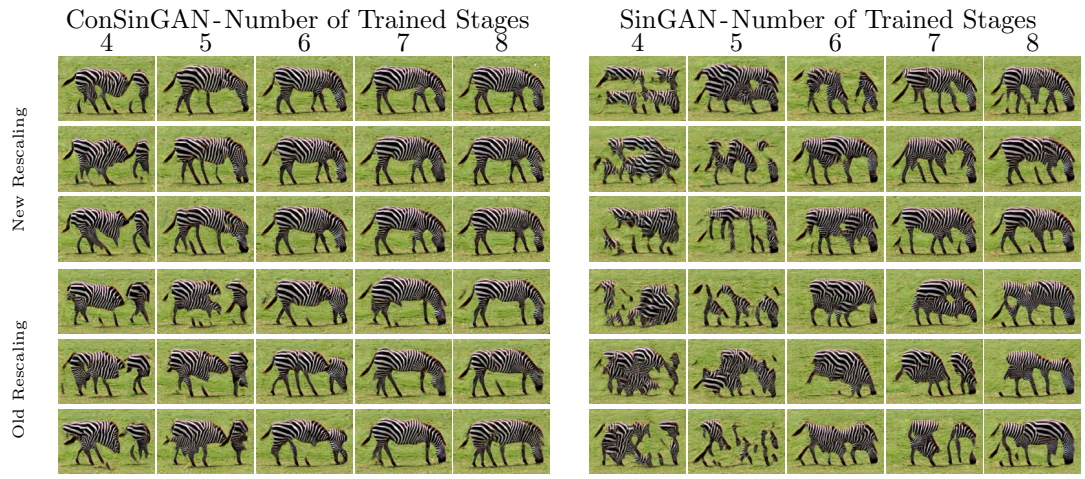


Figure 5.5: Comparison of the effect of the number of trained stages and rescaling method during training. Images are randomly selected.

Model	Random ↑	Paired ↑	SIFID ↓	Train Time	Stages	Params
ConSinGAN	$56.7\% \pm 1.9\%$	$63.1\% \pm 1.8\%$	0.11 ± 0.06	20 min	5.9	~660K
SinGAN	$43.3\% \pm 1.9\%$	$36.9\% \pm 1.8\%$	0.23 ± 0.15	135 min	9.1	~1.0M

Table 5.2: Results of our user studies and SIFID on images from the **LSUN** dataset.

appearance of the towers. However, with a $\delta = 0.1$, the appearance of the towers changes more drastically and sometimes even additional towers are added to the generated image. Unless otherwise mentioned, all illustrated examples and all images used for the user study where generated by models for which we trained three stages concurrently with $\delta = 0.1$.

Baseline comparisons We compare our model to the SinGAN [Shaham et al., 2019] model in Figure 5.6. For SinGAN, we show the results of both the default rescaling method (8-10 stages) and our rescaling method (5-6 stages). In the first example we observe that SinGAN struggles to model recurring structures (faces) in the generated images. In the second example we observe a loss of global structure independent of the number of stages trained. Our multi-stage training helps ensure a more consistent global structure.

Figure 5.5 further highlights the advantages of our approach by showing a detailed comparison of the images each model generates after being trained with the new or old rescaling technique. Each column depicts three randomly sampled images from each model. We can see the positive effect of the rescaling technique for both models, regardless of the number of trained stages. Furthermore, we can see that our model retains better global coherence in both cases.

Quantitative evaluation The Fréchet Inception Distance (FID) [Heusel et al., 2017] compares the distribution of a pre-trained network's activations between a

sets of generated and real images. The Single Image FID (SIFID) is an adaptation of the FID to the single image domain and compares the statistics of the network's activations between two individual images (generated and real). In our experiments, we found that SIFID exhibits very high variance across different images (scores range from $1e-06$ to $1e01$) without a clear distinction of which was "better" or "worse". In this work, we focus mostly on qualitative analyses and user studies for our evaluation but also report SIFID for comparison.

We performed quantitative evaluations on two datasets. The first dataset is the same as the one used by SinGAN, consisting of 50 images from several categories of the 'Places' dataset [Zhou et al., 2014]. However, many of these images do not exhibit a global layout or structure. Therefore, we also construct a second dataset, where we take five random samples from each of the ten classes of the LSUN dataset [Yu et al., 2015]. This dataset contains classes such as "church" and "bridge" which exhibit more global structures. We train both the SinGAN model and our model for each of the 50 images in both datasets and use the results for our evaluation.

Image Diversity We evaluate the diversity in our images compared to the original SinGAN model using the same measure as SinGAN: for a given training image we calculate the average of the standard deviation of all pixel values along the channel axis of 100 generated images. Then, we normalize this value by the standard deviation of the pixel values in the training image. On the data from the 'Places' dataset, SinGAN obtains a diversity score of 0.52, while our model's diversity is similar with a score of 0.50. When we increase the learning rate on lower stages by setting $\delta = 0.5$ instead of the default $\delta = 0.1$ we observe a lower diversity score of 0.43 as the model learns a more precise representation of the training image (Figure 5.4). On the LSUN data, SinGAN obtains a much higher diversity score of 0.64. This is due to the fact that it often fails to model the global structure and the resulting generated images differ greatly from the training image. Our model, on the other hand, obtains a diversity score of 0.54 which is similar to the score on the 'Places' dataset and indicates that our model can indeed learn the global structure of complex images.

Figure 5.6: Comparison of SinGAN and ConSinGAN.

User Study: 'Places' We follow the same evaluation procedure as previous work [Isola et al., 2017; Shaham et al., 2019; Zhang et al., 2016] to compare our model with SinGAN on the same training images that were used previously in [Shaham et al., 2019]. Users were shown our generated image and its respective

training image for one second each and were asked to identify the real image. We reproduced the user study from the SinGAN paper with our own trained SinGAN and ConSinGAN models. As we can see in Table 5.1 our model achieves results similar to the SinGAN model. However, our model is trained on fewer stages and with fewer parameters and obtains a better SIFID score of 0.06, compared to SinGAN's 0.09. Furthermore, the images generated by ConSinGAN often still exhibit a better global structure, but one second is not enough time for users to identify this.

User Study: 'LSUN' Since the images from the LSUN dataset are much more challenging than the images from the 'Places' dataset we do not compare the generated images against the real images, but instead compare the images generated by SinGAN to the ones generated by ConSinGAN. We generate 10 images per training image, resulting in 500 generated images each from SinGAN and ConSinGAN, and use these to compare the models in two different user studies.

In both versions, the participants see the two images generated by the two models next to each other and need to judge which image is better. We do not enforce a time limit, so participants can look at both images for as long as they choose. The difference between the two versions of the user study is how we sample the generated images. In the first version ("random") we randomly sample one image from the set of generated images of SinGAN and ConSinGAN each. This means that the two images likely come from different classes (e.g. 'church' vs. 'conference room'). In the second version ("paired") we sample two images that were generated from the same training image. We perform both user studies using Amazon Mechanical Turk, with 50 participants comparing 60 pairs of images for each study.

Table 5.2 shows how often users picked images generated by a given model for each of the two settings. We see that users prefer the images generated by ConSinGAN in both settings and that, again, our model achieves a better SIFID. This is the case even though our model only trains on six stages, has fewer parameters than SinGAN, and takes less time to train. The images from LSUN vary in difficulty and global structure. This might explain why our model performs even better in the paired setting since this setting guarantees that we always compare the two models on images of the same difficulty. Overall, our experiments show that ConSinGAN allows for the generation of more believable images, especially when they exhibit some degree of global structure, with less training time and a smaller model than SinGAN.

Image Harmonization

We now show results on image harmonization examples and compare our model to SinGAN and Deep Painterly Harmonization [Luan et al., 2018] for high-resolution images.

Training Details We train ConSinGAN with the same hyperparameters for all

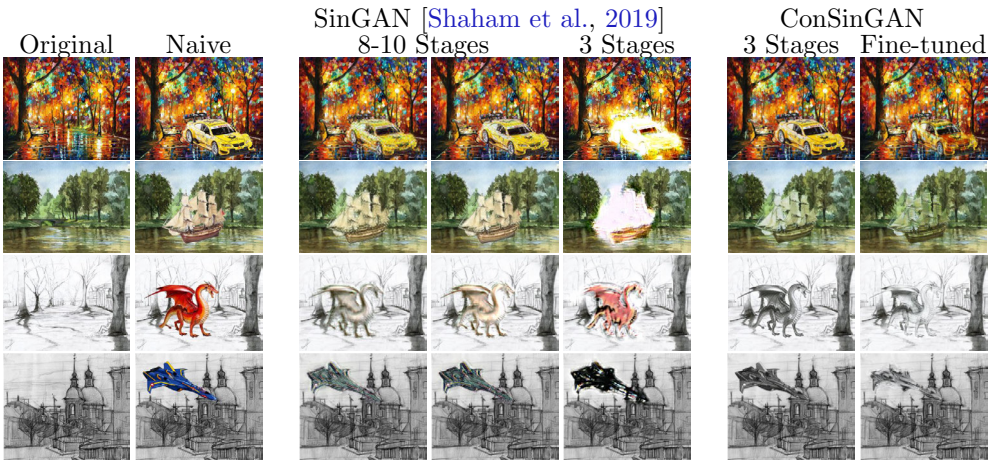


Figure 5.7: Image harmonization with SinGAN and ConSinGAN

images without any fine-tuning of hyperparameters for the different images. The general architecture is the same as for unconditional image generation, however, we only train the model for exactly three stages per image. We train for 1,000 iterations per stage and randomly sample from different data augmentation techniques to obtain a "new" training image at each iteration as described in Subection 5.2.3. When we fine-tune a model on a given specific image we use a model trained on the general style image and use the target image directly as input (instead of the style image with random augmentation transformations) to train the model for an additional 500 iterations.

Comparison with SinGAN Figure 5.7 shows comparisons between SinGAN and ConSinGAN. The first two columns show the original images we trained on and the naive cut-and-paste images that are the input to our trained model at test time. The next three images show the results of a trained SinGAN model, where the first two are the results of a fully trained model. We insert the naive image at all stages of the model and choose the two best results, while the third image is the result when we train SinGAN on only three stages. The final two columns show the results of the ConSinGAN. Training ConSinGAN takes less than 10 minutes for a given image when the coarse side of the image has a resolution of 250 pixels. Fine-tuning a model on a specific image takes roughly 2-3 minutes. Training SinGAN takes roughly 120 minutes as before, since we need to train the full model, even if only some of the later stages are used at test time.

We see that ConSinGAN performs similar to or better than SinGAN, even though we only train ConSinGAN for 3 stages. ConSinGAN also generally introduces fewer artifacts into the harmonized image, while SinGAN often changes the surface structure of the added objects. See for example the first row in Figure 5.7, where SinGAN adds artifacts onto the car, while ConSinGAN keeps the original objects consistent. When we fine-tune the ConSinGAN model on specific images we can get even more interesting results, as, e.g., the car gets absorbed much more into the colors of the overall background. The bottom two rows of Figure 5.7 show results

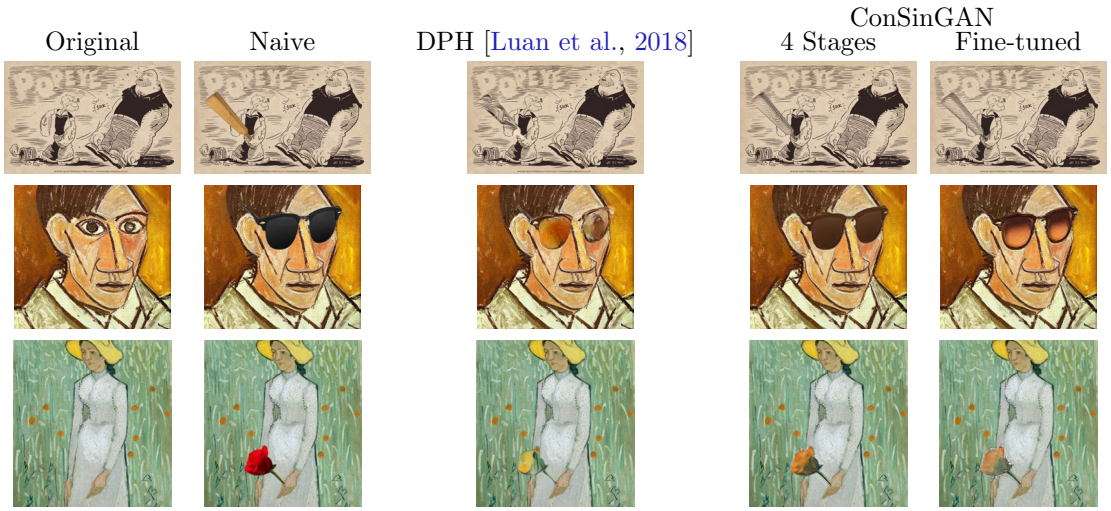


Figure 5.8: Image harmonization comparison with Deep Painterly Harmonization (DPH) and ConSinGAN on high resolution images

when we add colorful objects to black-and-white paintings. When training SinGAN on only three stages like ConSinGAN it usually fails completely to harmonize the objects at test time. Even the images harmonized after training SinGAN on 8-10 stages often contain some of the original colors, while ConSinGAN manages to completely transfer the objects to black-and-white versions. Again, further fine-tuning ConSinGAN on the specific images leads to an even stronger "absorption" of the objects.

Comparison with DPH Figure 5.8 shows comparisons between ConSinGAN, adapted to harmonize high-resolution images, and Deep Painterly Harmonization (DPH) [Luan et al., 2018]. The images have a resolution of roughly 700 pixels on the longer side, as opposed to the 250 pixels used by the SinGAN examples. In order to produce these high-resolution images, we add another stage to our ConSinGAN architecture, i.e. we now train four stages, and training time increases to roughly 30-40 minutes per image. This is in contrast to many style-transfer approaches and also DPH, which have additional hyperparameters such as the style and content weight which need to be fine-tuned for a specific style image.

We can see that the outputs of ConSinGAN usually differ from the outputs of DPH, but are still realistic and visually pleasing. This is even the case when our model has never seen the naive copy-and-paste image at train time, but only uses it at test time. In contrast to this, DPH requires as input the style input, the naive copy-and-past input, and the mask which specifies the location of the copied object in the image. Again, fine-tuning our model sometimes leads to even better results, but even the model trained only with random image augmentations performs well. While our training time is quite long, we only need to train our model once for a given image and can then add different objects at different locations at test time. This is not possible with DPH, which needs to be retrained whenever the copied object changes.

5.2.5 Conclusion

We introduced ConSinGAN, a GAN inspired by a number of best practices discovered for training single-image GANs. Our model is trained on sequentially increasing image resolutions, to first learn the global structure of the image, before learning texture and stylistic details later. Compared to other models, our approach allows for control over how closely the internal patch distribution of the training image is learned by adjusting the number of concurrently trained stages and the learning rate scaling at lower stages. Through this, we can decide how much diversity we want in the generated images. We also introduce a new image rescaling approach that allows training on fewer image scales than before. We show that our approach can be trained on a single image and can be used for tasks such as unconditional image generation, harmonization, editing, and animation while being smaller and more efficient to train than previous models.

5.3 Learning Representations for a Single Object from a Few Labelled Examples

This section presents our work *CharacterGAN: Few-Shot Keypoint Character Animation and Reposing* by Tobias Hinz, Matthew Fisher, Oliver Wang, Eli Shechtman, and Stefan Wermter.

5.3.1 Introduction

Modifying and animating artistic characters is a task that often requires experts to manually create many instances of the same character in different poses, which is a time-consuming and expensive process. Using video frame interpolation methods can reduce the overhead, but these methods do not leverage character-specific priors and so can only be used for small motions. Similarly, warping input frames directly with 2D handles cannot account for disocclusions or appearance changes between poses. In this work we have two main goals: (1) to generate high quality frames of an animated character based on a small number of examples, and (2) to generate these images based on a sparse set of keypoints that can be easily modified in real time.

To address both issues, we propose to train a conditional Generative Adversarial Network [Goodfellow et al., 2014] (GAN) architecture that allows us to create new images of a character based on a set of given keypoint locations. We show that, with the right form of implicit biases, such a model can be trained in a few-shot setting (i.e., 10s of training images). In character animation, each training image has to be created manually, making it expensive to acquire the number of images required for training most conditional GAN models [Isola et al., 2017], and unlike recent work on single image generative models [Shaham et al., 2019], we desire precise control over the generated image.

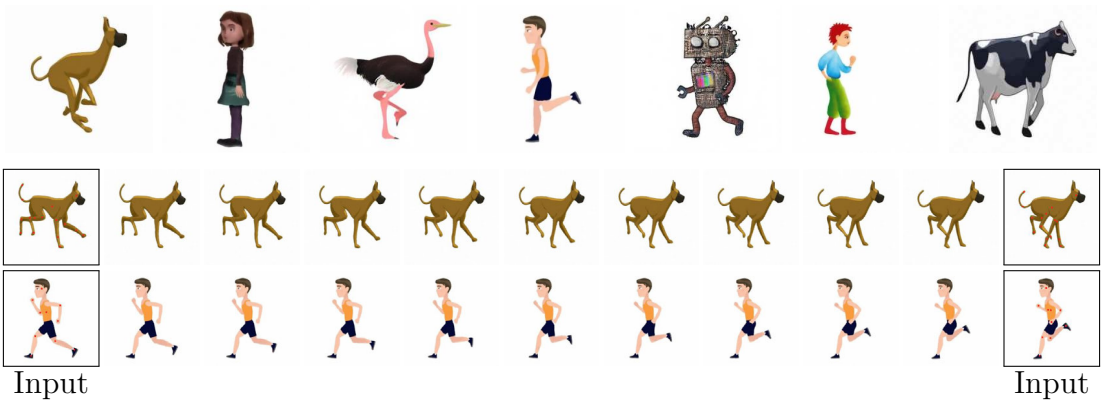


Figure 5.9: We train a generative model in a low-data setting (8 to 15 training samples) to repose and animate characters based on keypoint positions. The first row shows **video** results (indicated throughout the paper by 🎥), of our method driven by linearly interpolating input keypoints. Please view with Adobe Acrobat to see animations. The second and third rows show interpolated frames generated by our method between a *single* start and end frame (left and right columns)

Our method consists of GAN that is trained on 8 – 15 images of a given character and its associated keypoints in different poses. One of the key challenges is that it is difficult for the model to learn which parts of a character should be occluded by other parts. E.g. when the hand of a humanoid character moves in front of the torso, either the hand should be visible at all times (if it is in the foreground) or only the torso should be visible if the hand moves "behind" the torso. To learn this ordering in a purely data-driven approach we need many images which we do not have in our setting. Instead, we propose to use user-specified *layers* for our keypoints, i.e. each keypoint lies in a given layer and we can introduce an ordering over those layers. For example, a humanoid character could be described by three layers, consisting of 1) the arm and leg in the "back" (i.e. occluded by the torso and other arm and leg), 2) the torso and head which are occluded by one arm and one leg and occlude one arm and one leg, and 3) the other arm and leg which occlude every other part of the character. Our model processes each of these layers independently, i.e. generates features for each layer without knowing the location of the keypoints in the other layers. We then use an adaptive scaling approach, conditioned on all keypoints, to spatially scale the features of each layer, before concatenating and using them to generate the final image.

Additionally, we can train our generator to predict the mask for a generated character which we found to be a robust way to identify and automatically fix unrealistic keypoint layouts at test time. Our model naturally learns to associate the keypoints with roughly semantically meaningful body parts for each layer, and can handle discrete state points that arise as a function of keypoint locations (e.g., switching between a profile facing left or right). Finally, to improve image quality, we use a patch-based refinement step [Barnes et al., 2009] on the generated images based on [Jamriska, 2018].

We show via a number of qualitative and quantitative experiments that our resulting model allows for real-time reposing and animation of diverse characters and that our layering approach outperforms the more traditional conditioning approach of using all keypoints at once. Since we assume no prior knowledge about the modeled character our model can be applied to any shape and does not require additional data such as a 3D model or a character mesh. This allows our model to be applied in domains for which only limited data is available (e.g. artistic drawings or sprite sheets) without the need for additional manual input besides the keypoint labels. In summary, we introduce the following main contributions:

- We show that it is possible to train a GAN on only few images ($8 - 15$) of a given character to allow for few-shot character reposing and animation. By only conditioning the training on keypoints (instead of e.g. semantic maps) the trained model allows for character reposing in real-time without expert knowledge.
- By using a layered approach that explicitly encodes the ordering of different keypoints our model is able to model occlusions with only very limited training data.
- We introduce a mask connectivity constraint, where a jointly predicted mask can be used at test time to automatically fix keypoint layouts for which the model produces unrealistic outputs.

5.3.2 Related Work

Conditional GANs take as input some form of label which makes it possible to control the output of the generator to varying degrees and has also been shown to help with the training process. The label input can come in several forms, such as class conditioning [Mirza and Osindero, 2014], semantic maps [Isola et al., 2017], keypoints [Reed et al., 2016a], or bounding boxes [Hinz et al., 2019, 2020]. However, most conditional GANs are trained with large datasets and are applied to broader domains, whereas we are interested in animating a *specific* character. In the following, we first focus on approaches in how to leverage small amounts of data to train GANs and conclude with a section about how the chosen conditioning method directly affects how the model can be used at test time.

Few-Shot Learning with GANs One promising approach to few-shot learning with GANs is to use fine-tune GANs that are pre-trained on large datasets. This can be achieved by fine-tuning a given model [Wang et al., 2018b], by only training parts of the pre-trained model [Noguchi and Harada, 2019; Robb et al., 2020; Mo et al., 2020; Li et al., 2020b], or transfering knowledge between different - but related - domains [Wang et al., 2020; Zhao et al., 2020a]. However, these methods still rely on a model that is pre-trained on a large dataset in a very similar domain. As opposed to this we do not fine-tune a pre-trained model on a small dataset but instead train our model from scratch on limited available data.

Recent approaches show that applying data augmentation techniques during training of GANs directly is useful, especially when the dataset is small [Tran et al.,

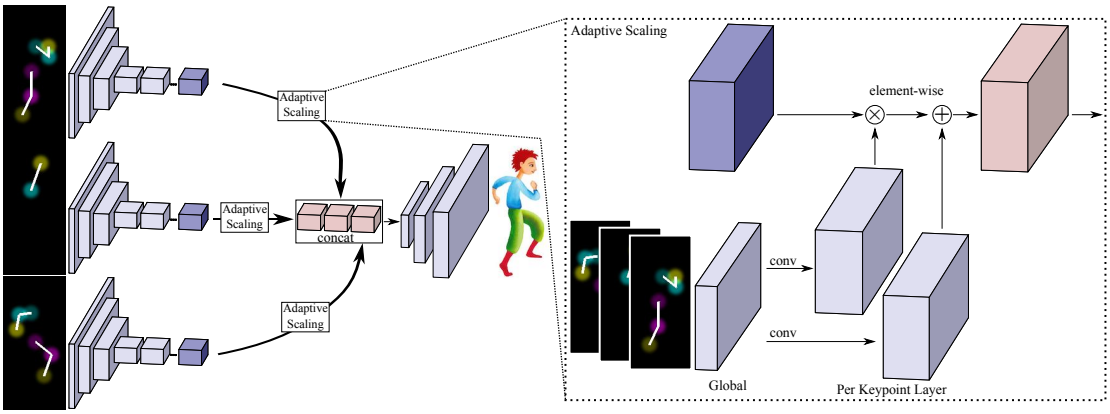


Figure 5.10: Our model processes keypoints which are split into individual layers. The resulting features for each keypoint layer are then scaled, concatenated, and used to generate the final image.

2020; Karras et al., 2020a; Zhao et al., 2020b,c]. One of the main insights of these approaches is that it is not sufficient to only augment real images since this leads the discriminator to learn that these augmented images are part of the real data. We also make heavy use of data augmentation in our training process but use much less data (8 – 15 images) than the approaches mentioned here (usually 100+ images).

It is also possible to learn useful features from only a single image [Zontak and Irani, 2011; Asano et al., 2020] and that tasks such as texture synthesis [Jetchev et al., 2016; Li and Wand, 2016; Bergmann et al., 2017; Zhou et al., 2018], image retargeting [Shocher et al., 2019], inpainting and segmentation [Ulyanov et al., 2018; Gandelsman et al., 2019], and unconditional image synthesis [Shaham et al., 2019; Hinz et al., 2021b] are possible with only a single image. Other approaches train GANs for image-to-image translation with only one pair of matching images [Lin et al., 2020; Benaim et al., 2020; Park et al., 2020; Vinker et al., 2020]. However, a model trained on just a single image without any other information can only generate limited variations of the training input. Therefore, we propose to increase the available information by slightly increasing the training set's size, which increases the model's capability to generate more variations of a learned object.

Editability An important characteristic of models for character reposing and animation is how the character is controlled. Existing approaches use driving videos [Siarohin et al., 2019], learn a distribution over poses for inbetweening [Poursaeed et al., 2020], or map puppets to skeletons [Dvorožňák et al., 2018]. However, it is difficult to achieve a desired pose exactly with these approaches [Siarohin et al., 2019; Poursaeed et al., 2020] or requires expert knowledge to obtain the required training data [Dvorožňák et al., 2018]. Another approach is to condition the model on a semantic label map [Chan et al., 2019; Vinker et al., 2020] which can be changed at test time to reflect the new layout. However, modifying semantic maps or edge maps is not something that can be done on-the-fly, but instead takes time and skill. Finally, one could start with the character in a given pose and warp or

stretch it until it reaches the desired pose [Liu et al., 2014] which might lead to unrealistic results if the end pose is too different from the starting pose. Given the previous limitations we only condition our model on high-level keypoints which are provided for a given character, making it easy and fast to generate new poses at test time by moving keypoints.

5.3.3 Methodology

In this section we describe our approach and how the model can be used at test time.

Input Requirements

In this work, we focus on character animation and reposing, where the images show the same character in different poses from the same viewpoint. Since our goal is to work in low data scenarios, we focus on cases with a small amount of images of the given character. Furthermore, the individual images can depict discrete appearance variations, e.g. different facial expressions. We do not make any other prior assumptions about the structure of the character. Our method requires an input image set, keypoints per image, keypoint connectivity information (essentially a skeleton), and a layer ordering for the keypoints. This information can be obtained easily as we only need labels for very few images. While the keypoints have to be labeled manually for each image, all other information is image independent and, thus, only needs to be defined once.

We experimented with various ways to input keypoints to the network, such as as individual channels, but found that representing keypoints as RGB Gaussian blobs performed the best. This also means that the input condition always has the same input dimensionality, independent of the number of keypoints, which is helpful as our model uses the same parameters to process these individual layers. Each keypoint is defined by a position $\{x, y\}$, three randomly chosen color values for the three RGB channels, and the values σ and r, where σ represents the falloff of the Gaussian distribution we use for blurring the keypoint, and r represent the radius of the final keypoint, which we define later. Discrete keypoints are modelled with individual RGB values, and in cases where different keypoints overlap we sum the respective colors.

Model

Our model consists of a generator and a discriminator (see Figure 5.10). The generator generates the image based on the keypoint locations while the discriminator is trained to distinguish between real and fake image-keypoint pairs. Finally, we apply a patch-match [Barnes et al., 2009] based refinement step to improve the final quality of the generated images.

Generator and Discriminator We base our architecture on pix2pixHD [Wang et al., 2018a]. However, while the discriminator is similar to the original pix2pixHD

discriminator, we add several implicit biases to the generator reflecting our prior knowledge about the problem. Concretely, we know that the modeled characters are inherently three-dimensional, i.e. if some body parts are occluded by others they still exist even though they may not be visible. To address this, we split our characters into different layers, e.g. representing the "left" side of the character (e.g. left arm and leg), the "middle" part of the character (e.g. head and torso) and the "right" side of the character (e.g. right arm and leg). These layers can be modeled individually and can then be composed to form a final image. To model this, our generator processes each keypoint layer individually and learns a representation of each keypoint layer (see Figure 5.10).

Intuitively, we could set some features to zero if they are occluded by other features. For example, if the left hand is "behind" the torso for a given image, zeroing out the features for the left hand might make it easier for the generator to generate a realistic image. Conversely, if the right hand is "in front of" the torso, zeroing out the torso features at the location of the right hand might improve the performance. To address this, we incorporate an adaptive scaling technique [Park et al., 2019] in which we scale the features of each layer before concatenating them. For this, we first learn an embedding of the keypoints and their layers ("Global" in *Adaptive Scaling* of Figure 5.10). Based on this embedding we then learn scaling parameters for each keypoint layer and use them to scale the features of each each layer. These scaled layer features are then concatenated and used to generate the final image.

Our discriminator takes the keypoint conditioning k concatenated with an RGB image as input and classifies it as either real or fake. We use two patch-discriminators [Isola et al., 2017], one of which operates on the full resolution image, while the second operates on the image down-scaled by a factor of two. We use a feature matching and an adversarial loss during training as defined by the pix2pixHD model [Wang et al., 2018a]. The adversarial loss is the standard GAN loss L_{adv}:

$$\begin{aligned} \min_G \max_D \mathcal{L}_{\text{adv}} = \mathbb{E}_{(k,x)}[\log D(k,x)] + \\ \mathbb{E}_{(k)}[\log(1 - D(k, G(k))] \end{aligned} \tag{5.4}$$

where k is the keypoint condition and x is the corresponding real image. The feature matching loss L_{fm} stabilizes training by forcing the generator to produced realistic features at multiple scales and is defined as:

$$\min_G \mathcal{L}_{\text{fm}} = \mathbb{E}_{(k,x)} \sum_i^T \frac{1}{N_i} [||D(k,x) - D(k, G(k))||_1] \tag{5.5}$$

where T is the number of layers in the discriminator and N_i is the number of elements in each layer. We add a perceptual loss [Johnson et al., 2016; Zhang et al., 2018b] to further improve the image quality. We use a VGG net to extract features from real and generated images and compute the perceptual loss L_{perc} as defined by [Johnson et al., 2016]. Our final loss is the combination of these losses:

$$\min_G \max_D \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{fm}} + \mathcal{L}_{\text{perc}}. \tag{5.6}$$

Dataset	SinGAN [Shaham et al., 2019]		ConSinGAN [Hinz et al., 2021b]	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
Watercolor Man	18.03 ± 0.09	0.149 ± 0.0032	21.65 ± 0.32	0.102 ± 0.0151
Watercolor Lady	18.03 ± 0.09	0.159 ± 0.0024	25.49 ± 0.14	0.070 ± 0.0012
Sprite Man	17.14 ± 0.08	0.156 ± 0.0063	23.19 ± 0.17	0.087 ± 0.0045
Dog	15.01 ± 0.28	0.195 ± 0.0078	19.24 ± 0.27	0.125 ± 0.0089
Ostrich	16.54 ± 0.03	0.200 ± 0.0023	23.30 ± 0.18	0.103 ± 0.0020
Cow	13.58 ± 0.03	0.220 ± 0.0016	18.71 ± 0.12	0.133 ± 0.0037

Dataset	DeepSIM [Vinker et al., 2020]		CharacterGAN	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
Watercolor Man	21.92 ± 0.03	0.066 ± 0.0002	**24.33 ± 0.05**	**0.042 ± 0.0002**
Watercolor Lady	26.21 ± 0.04	0.048 ± 0.0006	**28.27 ± 0.03**	**0.037 ± 0.0003**
Sprite Man	22.08 ± 0.07	0.071 ± 0.0001	**25.23 ± 0.02**	**0.038 ± 0.0006**
Dog	20.08 ± 0.07	0.087 ± 0.0010	**22.22 ± 0.04**	**0.062 ± 0.0006**
Ostrich	21.54 ± 0.13	0.079 ± 0.0006	**23.80 ± 0.09**	**0.063 ± 0.0005**
Cow	17.65 ± 0.03	0.115 ± 0.0013	**19.59 ± 0.01**	**0.085 ± 0.0005**

Table 5.3: Results of cross validation for the different models.

Patch-based Refinement To further improve the final result we apply a patch-based refinement algorithm that replaces generated patches with their closest real patch. In our case, given a real and a generated image, for each patch in the generated image, we find the closest patch in the dataset of all real images using the Patch Match approximate nearest neighbor algorithm [Barnes et al., 2009; Hertzmann et al., 2001], and replace the generated patches with their real equivalent [Texler et al., 2020]. We found that this approach often improves the sharpness and general image quality over the output of the generator (Figure 5.14).

Data Augmentation We employ both affine transformations and thin-plate-spline augmentation [Vinker et al., 2020]. We use a mixture of horizontal and vertical translations and horizontal flipping and randomly sample a subset from these augmentation approaches at each training iteration. Thin-plate-spline (TPS) augmentation was introduced by Vinker et al. [2020]. For this approach, the image is modeled as a grid and each grid point is then shifted by a random distance sampled from a uniform distribution. After this, a TPS is used to smooth the transformed grid into a more realistic warp. Using TPS augmentation results in warped images where parts of the image are stretched and elongated, adding further variation to the training data. All augmentations are applied to the given image and the associated keypoints and skeleton.

Dataset	CharacterGAN No Layer, No Scaling		CharacterGAN Layer, No Scaling	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
Watercolor Man	23.81 ± 0.15	0.049 ± 0.0050	24.29 ± 0.02	0.044 ± 0.0004
Watercolor Lady	28.23 ± 0.01	0.038 ± 0.0002	28.27 ± 0.05	0.037 ± 0.0003
Sprite Man	24.63 ± 0.12	0.041 ± 0.0020	25.12 ± 0.01	0.039 ± 0.0004
Dog	21.71 ± 0.10	0.068 ± 0.0007	22.14 ± 0.05	0.064 ± 0.0005
Ostrich	22.95 ± 0.04	0.068 ± 0.0007	22.97 ± 0.04	0.067 ± 0.0004
Cow	18.95 ± 0.08	0.094 ± 0.0021	19.52 ± 0.01	0.086 ± 0.0009

Watercolor Man Watercolor Lady	Sprite Man Dog	Ostrich Cow
		

Table 5.4: Ablation study: results of cross validation for different parts of our model.

Editability

After our model is trained it offers a straightforward way to modify the pose of the character. Given an image of the character the user can drag keypoints to novel positions and the generator will generate the character in the new pose. We can also easily switch between different discrete states, and we provide two ways to do this. First, it can be handled completely automatic where the discrete state is determined solely by keypoint positions (e.g., for facing left vs right). Second, we provide the ability to have specific keypoints for individual states, such as smile vs frown, so a user can choose the desired expression at inference time. We also allow the user to optionally enable a mask-based connectivity correction. In this case, if the user positions keypoints too far from their input distribution, such that they could lead to unrealistic or undesired results, we can automatically modify the keypoint locations to achieve more realistic results.

Ensuring Mask Connectivity If the image background is of uniform color (e.g. white), or we have a segmentation network, we can automatically extract a foreground-background mask. We can use this mask as additional conditioning information during training, i.e. in this case the generator does not only generate the RGB image but also the mask, while the discriminator gets as input an image and its associated keypoints and mask.

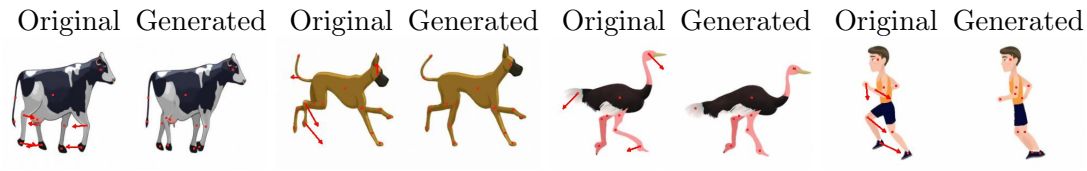


Figure 5.11: Examples from our model which was trained on only 8 – 12 images for each of the characters. Odd columns show the original image and our intended modifications, even columns show the output of our model.

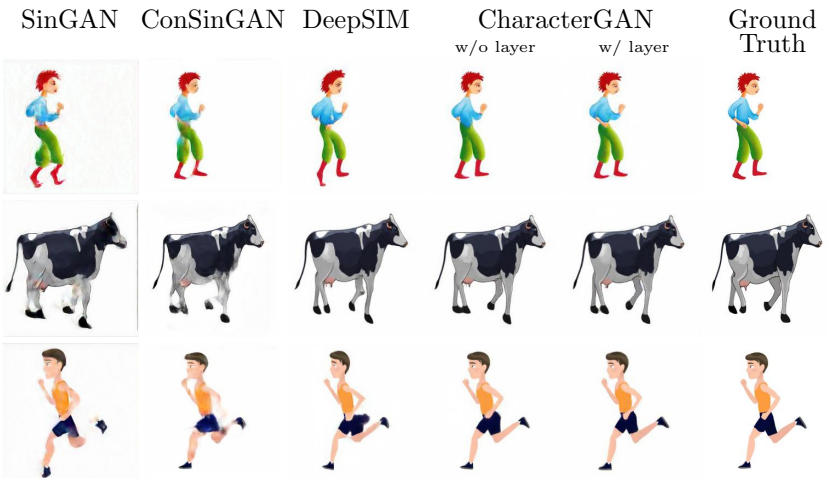


Figure 5.12: Qualitative examples of reconstructing held-out test images based on their keypoint locations.

At test time, if a keypoint is moved in a way that results in a layout that is too different from the ones seen at train time it can happen that the generator generates either "disconnected" body parts (e.g. the hand is not connected to the body) or introduces unwanted artifacts. Since the generator predicts the mask for the generated character we can use connected component analysis [Fiorio and Gustedt, 1996; Wu et al., 2005] to check whether the generated mask is connected. We found two cases that often lead to a disconnected mask; if the disconnected part contains a keypoint the resulting character will be ripped apart or, alternatively, if the disconnected part does not contain a keypoint the generated image will often contain unwanted artifacts. In either case, we fix the result by automatically moving nearby keypoints in an iterative procedure until the predicted mask is fully connected again.

Given the last moved keypoint that caused the a disconnected region to appear, we identify the closest keypoint and move this keypoint in the same direction as the keypoint moved by the user by a fraction δ of the absolute distance. If $\delta = 1$ the keypoint is moved exactly in the same direction and by the same amount as the original keypoint and if $\delta = 0$ no keypoints are moved automatically at all. We found that a default of $\delta = 0.5$ achieved good results in a single iteration in most cases, however, this process can be repeated iteratively until the mask is connected (Figure 5.15).

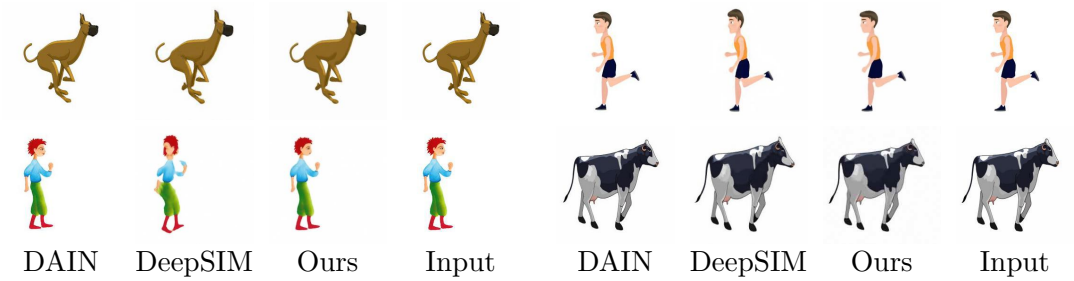


Figure 5.13: Comparison of our approach to frame interpolation [Bao et al., 2019] and DeepSIM [Vinker et al., 2020] using the displayed input frames (📹).

5.3.4 Experiments

Baselines To compare our approach we adapt three generative baseline models from the single-image domain to our setting. SinGAN [Shaham et al., 2019] and ConSinGAN [Hinz et al., 2021b] are trained on only a single image for tasks such as image generation and harmonization. Both models are trained in a multi-stage manner where the image resolution increases with each stage. While SinGAN trains each stage in isolation and freezes all previous stages, ConSinGAN trains the whole model end-to-end. We adapt both models by additionally conditioning the training at each stage on the image keypoints, i.e. each stage gets as input the keypoint condition in the respective image resolution. DeepSIM [Vinker et al., 2020] trains a model specifically for image manipulation, where the conditioning input consists either of edge-maps, semantic labels, or a combination of both. The model is trained on only a single instance of an image and its conditioning information. We adapt the DeepSIM model to our setting by replacing the edge-map conditioning with keypoints, i.e., instead of the edge-map the input to the model is a map of keypoint locations.

Experiments We perform experiments on several different characters, including humans and animals. Our data comes from different sources such as drawings by artists [Dvorožňák et al., 2018] and characters taken from sprite sheets. For our characters we have 8 – 15 images which we manually label with keypoint locations, but we also show that our model is able to handle larger datasets. Our model itself can generate images in real-time and takes about 0.01 seconds for one image on an NVIDIA RTX 2080Ti (0.4 seconds on CPU). One iteration of evaluating the mask connectivity takes about 0.00002 seconds on CPU and the patch based refinement takes about 0.6 seconds (1.8 seconds on CPU).

Quantitative Evaluation

To the best of our knowledge there are no established quantitative evaluation metrics for few-shot character animation. Shaham et al. [2019] introduced the Single-Image FID (SIFID) score to evaluate single-image generative models. However, Hinz et al. [2021b] report large variance in the SIFID scores and Robb et al. [2020] report that

models in the few-shot setting overfit to the FID metric. Other approaches use LPIPS [Zhang et al., 2018b] to compare a generated image with its ground truth counterpart. We use the Peak Signal-to-Noise Ratio (PSNR) and LPIPS as metrics to evaluate our model.

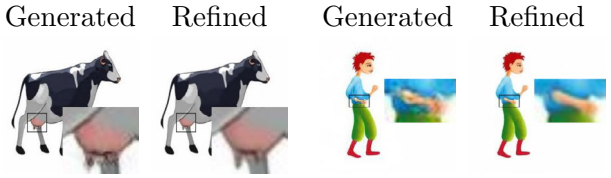


Figure 5.14: Effects of patch-based refinement (please view on screen and zoomed in).

To evaluate our model we design an N-fold cross-validation for a given character with N images. Given a character we train our model N times on $N-1$ images were each image in the dataset is left out of training exactly once. At test time we generate the left-out image based on its keypoint layout and calculate the PSNR and LPIPS between the generated and ground-truth image. For each character we run the full N-fold cross-validation three times and report the average and standard deviation across the three runs. We perform all our quantitative evaluations without using the patch refinement step to evaluate the models directly. Some examples are shown in Figure 5.12.

Table 5.3 shows the results of our model compared to the baselines. Our model achieves the best LPIPS and PSNR for all characters. We observe that the PSNR is not always predictive of the (perceptive) quality of the generated image. In particular, SinGAN and ConSinGAN often generate images where the character exhibits disconnected body parts (e.g. the feet are not connected to the main body), but this is not represented in the PSNR, as feet and legs cover a relatively small area of the image.

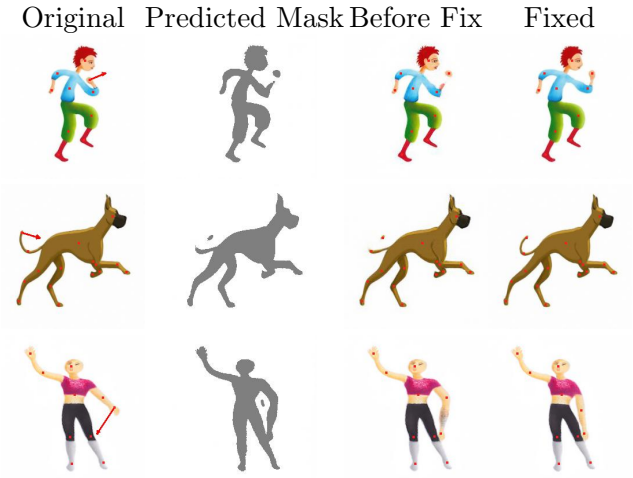


Figure 5.15: Enforcing mask connectivity at test time results in more realistic images.

Table 5.4 shows ablation studies with our model, where we train the same model without any layering, and with layering but no adaptive scaling. We can see that the layering approach improves the performance in all cases but for the "Watercolor Lady" character. This character is the only character of these that does not include occlusions, i.e. none of the bodyparts overlap each other which supports our hypothesis that the layering approach is mainly helpful for modeling occlusions in the low-data setting. Finally, adding the adaptive scaling further improves the performance, albeit not as much as the keypoint layering.

Qualitative Evaluation

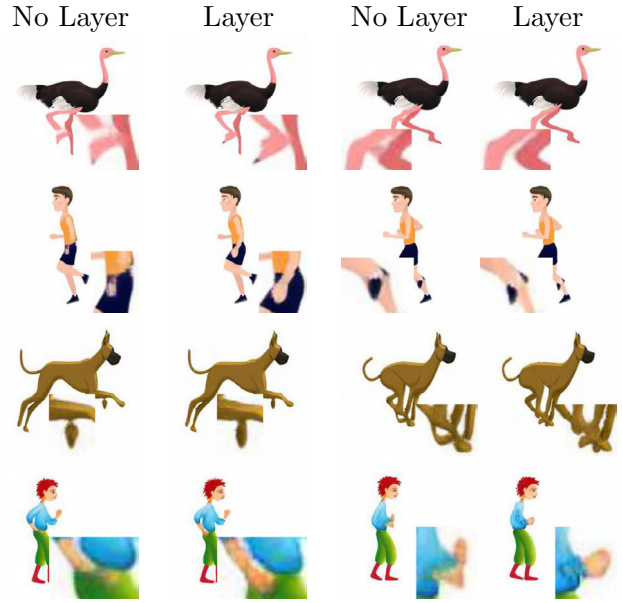


Figure 5.16: Comparison of layered vs non-layered at occlusions/overlaps.

Figure 5.11 shows visualizations of our model's capabilities on several different characters. All examples in this figure are trained on sprite sheets which contain 8 – 12 examples of the given character in different poses. Our model learns to generate realistic samples of four-legged animals, two-legged animals, and humanoid shapes. Furthermore, we see that our model can handle the movement of keypoints that relate to relatively small character parts (e.g. individual feet) as well as keypoints that represent large body parts (e.g. head and torso). Even when we move multiple keypoints, the resulting image is realistic, adheres to the novel keypoint layout, and leaves areas of the character that were not modified unchanged. Figure 5.14 shows the effect of the patch-based refinement algorithm which often improves small details of the generated images.

Figure 5.9 and Figure 5.13 show how our model can be used for character animation. We use the poses from the training set as starting point and linearly interpolate the keypoints to generate the intermediate frames. Our model produces smooth and realistic results. We note that these intermediate keypoint locations could also be derived from other data e.g. by extracting keypoints from driving videos [Aberman et al., 2018]. For comparison, we also show the results of a recent general purpose frame-interpolation model DAIN [Bao et al., 2019].

Figure 5.17: Visualization of what different layers learn (📹).

Mask-based Keypoint Refinement When moving certain keypoints to a new location we sometimes observe that this leads to "rips" in the generated character since the keypoint is too far away from the main body and such a configuration does not occur in training data. This can usually be fixed by moving the connected body part in the same direction as the original keypoint. Figure 5.15 shows examples of rips in the generated character (first two rows) and introduced artifacts (third row). The first column shows the original image and user specified modifications. The second and third columns show the predicted mask and generated image. The last column shows our model's final output after

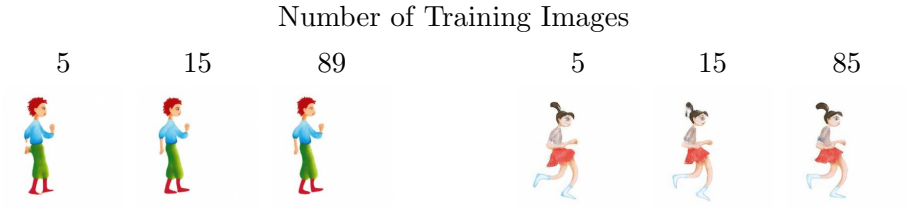


Figure 5.20: Here we show how performance improves with more training images (📹).

the respective connected keypoint was moved automatically. We can see that by enforcing the mask connectivity constraint we can get more realistic results in all cases.

Layered vs Non-layered Architecture Figure 5.16 shows qualitative comparisons between images generated by our architecture while using either our layered approach or a traditional, non-layered approach. As discussed previously, our layered approach is beneficial when several keypoints overlap in the 2D space, e.g. if a hand passes in front of a body or if two body parts are very close to each other. Figure 5.17 visualizes the features that our model learns for each keypoint layer. As we can see the model learns to only model the relevant keypoints and their associated body parts for each layer.

Automatic Appearance Switching Our model does not only learn to associate discrete keypoints with given features, but also learns to associate different features with a keypoint based on its location relative to other keypoints. Figure 5.18 shows several examples in which we can see that individual parts get "flipped" as a function of the location of that keypoint with respect to the others. As in our previous examples, the features of unrelated keypoints are unaffected by this.

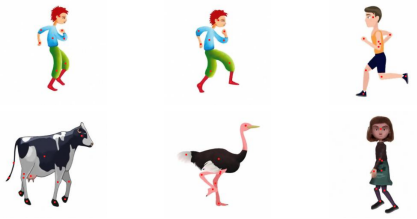


Figure 5.18: Discrete appearance changes based on keypoint location (📹).

Appearance-specific Keypoints Figure 5.19 shows how our model is able to switch between discrete appearance states for given keypoints. Each of the characters shows different visual features during training which we encode as different keypoint conditions. At test time we can switch between these different states to combine novel poses with any of the discrete visual expressions. Note that, again, our model learns to associate good features with the given keypoints, allowing us to model the poses independently of the discrete keypoint states.

Scaling to Larger Datasets While we show that our model performs well with only $8 - 15$ training images, we also evaluate our model on characters for which we have more training images (≥ 50). Figure 5.20 shows how our model scales with larger datasets. We see that more data is especially helpful when there are overlaps and occlusions. While the models trained on only 5 or 15 images have

difficulty modeling this (e.g. when the hand moves in front of the body), the models which are trained on more images perform much better.

Limitations Our model addresses the main challenge of correctly modeling (dis)occlusions based on limited training data. However, our model still has no explicit understanding about any underlying 3D representation of the character and Figure 5.20 shows how modeling occlusions gets worse with fewer training images. This could be addressed by future work, e.g. by adding known character priors into the model or by incorporating some form of 3D understanding.

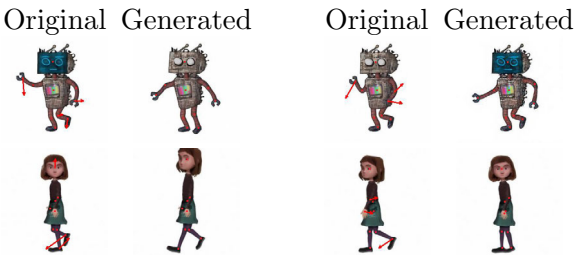


Figure 5.19: Discrete appearance change based on keypoint selection. In this example, the user not only moves keypoints to generate a new pose, but switches the IDs of keypoints to change expression, such as the color and rotation of the characters' faces.

5.3.5 Conclusion

In this work we show how to train GANs on few examples (8 – 15 images) of a given character for few-shot character reposing and animation. The model is easy to use, requires no expert knowledge, and our layering approach produces realistic results for novel poses and occlusions. We can perform character reposing in real-time through moving around keypoints and can animate character by interpolating keypoints. In the future, this can be used with other approaches, e.g. by extracting keypoints from driving videos for character animation. Through the use of a predicted foreground mask we can also automatically fix keypoint layouts that lead to unrealistic character poses. Finally, we show that our model learns discrete state changes based on keypoint locations, associates keypoints and their layers with semantic body parts, and scales to larger datasets.

5.4 Intermediate Discussion

We showed that it is possible to learn a useful representation for a single object from only a single data point Section 5.2. By training a GAN on a single image we can use the resulting representation for tasks such as unconditional image generation, image harmonization, image editing, and many more. However, training a model on only a single data point might be overly restrictive if we have more than one data point available. In Section 5.3 we show that we can use a small number of data points to learn good representations of a given object (in our case characters) for tasks such as reposing and animation. These approaches can be applied to many domains and are especially useful when we only have limited training data available. Additionally, we assume no previous knowledge about the task or underlying data but instead learn everything from scratch purely from the available data. This makes it easy for users without domain specific knowledge to apply these approaches

and obtain good results without fine-tuning or domain-specific design. However, our approaches still have two main shortcomings: they only learn a 2D representation of the underlying objects and they do not incorporate external knowledge into the learning process. We now talk about these two challenges and how they could be addressed in the future and how our current approaches could be combined with compositional representations (Section 4.1).

2D versus 3D Representation Our introduced approaches are based on GANs built with convolutional layers and are only trained with 2D images from a single viewpoint. As such, our models develop no 3D understanding of the objects they model. This makes it difficult to model certain aspects (e.g. occlusions in Section 5.3) and limits the application to tasks such as relighting and changing perspectives. An important future direction is, therefore, to incorporate 3D knowledge into our generative models. This could be done by either training the models on a 3D representation of the input data (e.g. point clouds or voxels [Sitzmann et al., 2019a]) or by adding certain inductive biases to the model architecture as done in implicit models [Mildenhall et al., 2020].

One promising direction is to combine implicit models with GANs. Traditional GANs can generate high-resolution images, are good at modeling textures, and are fast at generating new images at test time. However, they often overfit to textures instead of object shapes [Geirhos et al., 2019; Hinz et al., 2020] and are usually trained on 2D data with no inductive bias towards modeling the data from a 3D perspective. Implicit models, on the other hand, have an inductive bias towards modeling objects in a 3D manner and can model complex object shapes [Zhang et al., 2020]. However, implicit models are much slower at generating an image at test time, most current implicit models are only trained on synthetic data sets, and learning a general representation to quickly learn individual objects is still in its infancy [Sitzmann et al., 2020a; Tancik et al., 2020].

There are some early approaches at combining the advantages of implicit and traditional convolutional models. All of them focus on incorporating a 3D inductive bias into the generator while the discriminator is a 2D convolutional neural network as in traditional GANs. Schwarz et al. [2020] and Chan et al. [2020] replace the generator with an implicit model which learns a 3D representation from which the final 2D image is generated. Niemeyer and Geiger [2020] also use an implicit model in the generator, but only generate a low resolution 3D volume which is then used by a convolutional network to render the final image. However, these novel approaches often rely on synthetic data sets or on data that comes from a single domain, e.g. faces or cars, and have difficulty in modeling complex image backgrounds. More work is needed to improve the performance of these models to more complex data sets, e.g. by a tighter integration between convolutional and implicit models in the generator.

Incorporating External Knowledge One of the main drawbacks of learning a model from very little training data is that the model is constrained in the amount of knowledge it can learn. E.g., when the model is trained on only a single image of a human character it can only learn low-level features but no semantic

knowledge. As a consequence, the model will not understand the difference between e.g. an "arm" and a "head". However, if the model had a general understanding of a humanoid body it would be able to learn a much better representation, even if only provided with little training data. One important future direction is, therefore, to examine different approaches of how we can incorporate external knowledge about a given object or concept while training or fine-tuning a model on a given instance of that concept or object.

There are several possible ways this could be achieved. One approach could be to fine-tune existing models on a given object. This could be achieved by using meta-learning to learn a good "base" model for a given domain and then fine-tune or train this model on a given instance of the domain. One could also train a model on a large-scale data set to develop a general "understanding" of the context before adapting it to the given object. Another direction would be to incorporate strong inductive or explicit biases into the model architecture. These could be of general nature, e.g. 3D representations and object permanence or they could be more specialized and contain specific domain knowledge. Whatever the final approach will look like exactly, adding external semantic knowledge to models trained on very little data will likely drastically improve their performance and applicability to many different tasks.

Combination with Compositional Representations In the future, these learned representations of a single given object could naturally be combined with compositional representations. There are already some approaches that go into this direction. Niemeyer and Geiger [2020] use an implicit model as generator and use the implicit model iteratively to generate features for different objects. These objects are then rendered to a 2D image. Guo et al. [2020] use a purely implicit model to render more complex scenes that consist of individually controllable objects. Different implicit models are trained on the individual objects. However, these approaches are still limited to either quite simple data [Niemeyer and Geiger, 2020] or have strong data requirements and need a large amount of individual object training data [Guo et al., 2020]. We believe that exploring ways in which we can learn object specific representations in an efficient manner and use these representations to learn good and generalizable compositional representations are an important future direction of research.

5.5 Summary

We showed that it is possible to learn useful representations from a very limited number of training samples. Even if, in the limit, we only have a single data point for training we can still learn a representation that facilitates tasks such as image generation and harmonization (Section 5.2). If we scale this to a slightly larger training data set (e.g. 15 data points), the results become even better and open up applications such as character reposing and animation (Section 5.3). Being able to learn good representations makes it possible to apply these models in domains were large data sets are difficult to obtain, e.g. for specific artistic styles, manual

drawings, or in the medical domain. We also address several shortcomings, e.g. not incorporating 3D biases and the lack of external semantic knowledge, and suggest approaches how these shortcomings can be addressed in the future. Finally, we believe that an integration of compositional representations with representations specifically for individual objects is a promising future research direction that should be explored in more detail.

As alluded to in Chapter 3 and Chapter 4, these object-specific representations can potentially be useful for many other tasks and representations. For example, by making the representations highly specific we can define disentanglement on a per-object basis instead of having to find a generally applicable definition. Object-specific representations can also be combined with compositional representations to further increase their capabilities and generalization qualities. In the future, object-specific representation could be used as building blocks of more complex models, allowing for more flexibility and potentially making them extensible to novel objects when they are required.

Chapter 6

Conclusion

Learning good representations of data is one of the most crucial tasks of machine learning. Without a good representation, many tasks can not be solved efficiently, knowledge can not be transferred between domains, and it is difficult to extract and understand learned knowledge. Much of the success of current deep learning approaches is based on the fact that these approaches learn good, hierarchical representations that are useful for many tasks. This thesis expands on current approaches to use generative models to learn representations that possess certain properties, including disentanglement and compositionality, deemed useful for many tasks.

6.1 Thesis Summary

This thesis addressed three characteristics that we would like representations to possess. We now summarize our contributions based on the original research questions posed in Chapter 1.

Disentangled representations (Chapter 3): How can we learn disentangled representations with GANs? We show that we can learn disentangled representations without or with very few labels by maximizing the mutual information between a learned representation and the training data. One of the key decisions is how to structure the representations and what kinds of prior distributions are assumed for the disentanglement. Here, we divide our representations into two parts: the first part is used to model underlying noise and variations that do not follow a clear structure. The second part models the data generating factors and is learned by maximizing the mutual information between itself and the training data. We show that we can use both continuous and discrete distributions to learn different forms of data generating factors.

Our experiments show that we can learn many different underlying data generating factors, such as class labels, background characteristics, colors, etc. The learned data generating factors are partly – but not always – interpretable by humans. We evaluate our disentangled representations qualitatively by visualizing the images associated with specific representations. Additionally, we perform quantitative

evaluations by using pre-trained networks and show that specific changes in the disentangled representations correspond to specific changes in the associated images.

Compositional representations (Chapter 4): Can GANs learn compositional representations that decompose the underlying distribution? We introduce approaches to learn compositional representations with GANs. These compositional representations explicitly model individual objects based on class labels and additional characteristics such as size and location. By training these models end-to-end on large data sets we learn good representations for a large set of distinct objects. Through these individual representations, we can control the position and identity of individual objects, allowing for the modeling and generation of more complex scenes than previously possible.

Our experiments show that these representations generalize to novel settings. We perform several studies on controllable (synthetic) data sets and show that the model can generalize to modeling objects at locations at which it has not seen them during training, can model more or fewer objects than seen at train time, and can also model objects at sizes not seen during training. Furthermore, the model even generalizes to novel object characteristics, e.g. modeling objects in a color it has only observed on other objects at train time.

We introduce a novel evaluation metric – Semantic Object Accuracy (SOA) – that evaluates how well individual objects are modeled in a larger context. For this, we evaluate whether objects that should be in a given generated image are actually detected by a pre-trained object detector. Based on this score we identify a number of different characteristics that help when modeling individual objects, such as object size and texture, and also illustrate that modeling objects explicitly leads to better results than modeling objects implicitly.

Object-specific representations (Chapter 5): Can we use GANs to learn representations specifically for a single object from limited data? Our experiments show that we can learn good representations for specific tasks such as image harmonization and editing from only a single training data point. If we have a few more training samples (e.g. 15) we can learn representations of a specific object that allow us to achieve good results on tasks such as reposing and animation. These results highlight the power of learning object-specific representations that focus all their capacity on modeling a single specific object.

We identify several important approaches to learn good representations from only limited data. One approach relies on the limited receptive field of a convolutional neural network. By restricting the size of the receptive field of the discriminator and training the model on several different image resolutions we can learn good representations of individual image patches without overfitting to the image. Using these image patches we can generate new images or modify a given one. Another approach relies on specific kinds of data augmentation to artificially increase the size of the training data set. We show that both approaches lead to good results when applied correctly and in well-chosen domains.

6.2 Discussion and Future Research Directions

This thesis introduced several approaches to learn good representations with Generative Adversarial Networks (GANs). While we made progress in several important directions there are still many open questions and research directions. We now discuss some of the limitations of our current approaches and how they could be addressed in the future.

What Makes a Good Representation We identified a number of properties that make a representation "good" for specific tasks. Our approaches demonstrate how we can learn disentangled representations with minimal supervision in specified domains. Enforcing disentanglement allows us to extract previously unknown knowledge about the data generating factors from the learned representations, enables fine-grained control over the data generating process, and allows for interpretable representations. Furthermore, we show how the use of compositional representations enables us to model much more complex environments while giving us explicit control over individual objects and their various characteristics such as location and shape. Finally, object specific representations allow us to spend all of the representation's capacity on modeling a single individual object, making it possible to learn meaningful representation from only very little training data. We show the beneficial effects of these three representation characteristics in isolation, however, there is still more to be done in order to obtain "good" representations.

Bengio et al. [2013] propose many more characteristics that might be beneficial. However, different characteristics might be more or less important depending on the task the representations are used for, the data they are trained for, and the model architecture that is used. Many current approaches focus specifically on a few properties and set out to learn representations that possess these and perform well on a chosen task. Relatively little research exists that actually explicitly evaluates different representation properties and their effect on different tasks. Moreover, even if different properties are evaluated for their impact, there exist hardly any studies that evaluate how different combinations of representation characteristics affect each other. This is partly due to the fact that most representation learning approaches only focus on one specific aspect or characteristic of a representation and partly due to the fact that evaluating second or third-order interactions between representation properties becomes increasingly difficult. This is further exacerbated by the many different and sometimes conflicting definitions of popular characteristics such as disentanglement. To address this, future work should address both the clear definition of specific representation properties and how to evaluate those properties quantitatively across different domains.

Combine Our Approaches For all that we criticized in the previous paragraph, we actually did exactly that. We look into three different approaches for representation learning and evaluate them in isolation on tasks that are appropriate for the given representation characteristics. However, all of our approaches are conceptually "easy" to combine. Our final approach deals with learning object-specific representations from little data. The learned representations are highly

specialized for a given object. These approaches could, of course, be extended by or combined with our approaches to learn disentangled representations. Through this, we could get disentangled object-specific representations. We could even define "disentanglement" differently depending on the kind of object we are learning. This could help address the difficulty in defining disentanglement, since we would not need to define and evaluate it on a global level, but could use it on an object specific level with a definition and evaluation specifically suited to the object we are currently working with.

Once we have disentangled object-specific representations we could of course combine several of them to create compositional representations that can model environments that consist of these different kinds of objects. We could also use a different notion of disentanglement to learn compositional disentangled representation where disentanglement could be modeled on an object level. Alternatively, we could apply compositional representations to learning object-specific representations by using the compositional approach to learn part-based representations of a given object. As we see, there are many different ways in which our approaches can be combined. We can even combine them multiple times by applying the different concepts on a part-based, object-based, and scene-based level. Future work should look into all of these possibilities and how they might further improve the quality of learned representations.

6.3 Conclusion

In conclusion, this thesis contributes several approaches for improved representation learning with Generative Adversarial Networks (GANs). We evaluate several implicit biases and training procedures to incorporate disentanglement, compositionality, and object specialization into the learned representations. Our results show that learning disentangled representations is possible with only minimal supervision as long as we are confined to a specific domain where we can clearly define disentanglement. Once these conditions are met our approaches learn meaningful representations were different – interpretable – data generating factors are encoded in distinct parts of the representations. Furthermore, we highlight the limitations of current disentangled representations in the context of complex environments comprising several different objects. To approach this problem we develop an approach for learning compositional representations in which different objects are encoded explicitly. We show that our approach is able to learn representations for complex environments that are difficult to model with traditional, i.e. non-compositional representations. Finally, we develop models that can learn object-specific representations. These models spend all their representation capacity on learning the characteristics of a single object or concept. Through this, we are able to learn powerful but specific representations from only very limited training data without any previous domain knowledge. We evaluate all our approaches on several computer vision and image generation tasks and show that they lead to meaningful improvements compared to current state-of-the-art baselines. Future work should

look into combining several of these characteristics to learn even more powerful representations and examine approaches to incorporate implicit biases for the real world 3D structure into generative models.

Appendix A

Publications Originating from this Thesis

The following publications – listed in chronological order – were used directly for this thesis.

- **Hinz, T.**, Wermter, S. (2018). Inferencing based on unsupervised learning of disentangled representations. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (pp. 61-66).
  Used for Section 3.2.

- **Hinz, T.**, Wermter, S. (2018). Image generation and translation with disentangled representations. In *IEEE International Joint Conference on Neural Networks* (pp. 5519-5526).
  Used for Section 3.3.

- **Hinz, T.**, Heinrich, S., Wermter, S. (2019). Generating multiple objects at spatially distinct locations. In *International Conference on Learning Representations*.
  Used for Section 4.2.

- **Hinz, T.**, Heinrich, S., Wermter, S. (2020). Semantic object accuracy for generative text-to-image synthesis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, doi: 10.1109/TPAMI.2020.3021209.
  Used for Section 4.3.

- **Hinz, T.**, Fisher, M., Wang, O., Wermter, S. (2021). Improved Techniques for Training Single-Image GANs. *Winter Conference on Applications of Computer Vision* (pp. 1300-1309).
  Used for Section 5.2.

- **Hinz, T.**, Fisher, M., Wang, O., Shechtman, E., Wermter, S. (2021). CharacterGAN: Few-Shot Keypoint Character Animation and Reposing. *arXiv preprint arXiv:2102.03141*.
  Used for Section 5.3.

Other publications – listed in chronological order – were not used directly for this thesis:

- **Hinz, T.**, Navarro-Guerrero, N., Magg, S., Wermter, S. (2018). Speeding up the hyperparameter optimization of deep convolutional neural networks. In *International Journal of Computational Intelligence and Applications, 17, 02.*
- Soll, M., **Hinz, T.**, Magg, S., Wermter, S. (2019). Evaluating defensive distillation for defending text processing neural networks against adversarial examples. In *International Conference on Artificial Neural Networks* (pp. 685-696).
- Mao, J., Yao, Y., Heinrich, S., **Hinz, T.**, Weber, C., Wermter, S., Liu, Z., Sun, M. (2019). Bootstrapping knowledge graphs from images and text. In *Frontiers in Neurorobotics, 13, 93.*
- Heinrich, S., Yao, Y., **Hinz, T.**, Liu, Z., Hummel, T., Kerzel, M., Weber, C., Wermter, S. (2020). Crossmodal language grounding in an embodied neurocognitive model. In *Frontiers in Neurorobotics 14, 52.*
- Frolov, S., **Hinz, T.**, Raue, F., Hees, J., Dengel, A. (2021). Adversarial text-to-image synthesis: a review. *arXiv preprint arXiv:2101.09983.*

Appendix B

Supplementary Material

B.1 Learning Disentangled Representations

This section gives more details about our experiments outlined in Section 3.3.

B.1.1 Implementation Details

The generator is implemented as a deconvolutional neural network, while the encoder and the discriminator are convolutional neural networks. For the SVHN and the CelebA data sets we use the same architecture, while we use slightly smaller networks for the MNIST data set. For an overview of the used architectures see Table B.1.

In our experiments the weight λ_1 for the supervised loss was set to 10 to ensure that the encoder learns the labeled data correctly, the weighting λ_2 of the reconstruction loss was set to 1, and λ_3 and λ_4 were linearly increased from 0 to 1 during the first 1000 (10000) iterations on the MNIST (SVHN, CelebA) data set. We train the model for 50000, 150000 and 300000 iterations respectively on the MNIST, the SVHN, and the CelebA data set. The learning rate is 0.0001 for the discriminator and 0.0003 for the generator and the encoder, and the batch size is 64 in all experiments. For training, we use the Adam optimizer [Kingma and Ba, 2015] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

Since we only use a small amount of labeled data, we initially favor drawing labeled samples from the training set. In the beginning of the training process the probability of drawing labeled data is therefore one. During the first 1000 (10000) iterations on the MNIST (SVHN, CelebA) data set this probability is linearly decreased until it reaches the actual labeled sample ratio in the data set. For our latent representation, u is sampled from a uniform distribution $U(-1, 1)$, while c is split up into categorical and continuous variables c_{cat} and c_{cont}. For the categorical variables we use the softmax activation in the final layer, while the continuous variables are modeled as a factored Gaussian.

Table B.1: Overview of our network architectures.

MNIST	SVHN / CelebA
Discriminator	
dropout with probability 0.3 after each layer	
image X	
3x3 conv. 64, BN, ELU, stride 2	4x4 conv. 64, BN, ELU, stride 2
3x3 conv. 128, BN, ELU, stride 2	4x4 conv. 128, BN, ELU, stride 2
	4x4 conv. 256, BN, ELU, stride 2
FC 512, BN, ELU	FC 1024, BN, ELU
representation Z	
1x1 conv. 64, BN, ELU, stride 1	1x1 conv. 64, BN, ELU, stride 1
1x1 conv. 128, BN, ELU, stride 1	1x1 conv. 128, BN, ELU, stride 1
	1x1 conv. 256, BN, ELU, stride 1
FC 512, BN, ELU	FC 1024, BN, ELU
concatenate across channel axis	
FC 1024, BN, ELU	FC 1024, BN, ELU
FC 1, Sigmoid	FC 1, Sigmoid
Generator	
FC 3136, BN, ELU	FC 2048, BN, ELU
reshape 7x7x64	reshape 4x4x128
4x4 deconv. 128, BN, ELU, stride 2	4x4 deconv. 128, BN, ELU, stride 2
4x4 deconv. 64, BN, ELU, stride 1	4x4 deconv. 64, BN, ELU, stride 2
	4x4 deconv. 32, BN, ELU, stride 2
4x4 deconv. 1, Sigmoid, stride 2	3x3 deconv. 3, Sigmoid, stride 1
Encoder	
3x3 conv. 32, BN, ELU, stride 1	3x3 conv. 32, BN, ELU, stride 1
3x3 conv. 64, BN, ELU, stride 2	3x3 conv. 64, BN, ELU, stride 2
3x3 conv. 128, BN, ELU, stride 2	3x3 conv. 128, BN, ELU, stride 1
	3x3 conv. 256, BN, ELU, stride 2
	3x3 conv. 512, BN, ELU, stride 2
FC 1024, BN, ELU	FC 1024, BN, ELU
FC output layer	FC output layer

B.2 Learning Compositional Representations

This section gives more details about our experiments outlined in Section 4.2 and also shows additional results.

B.2.1 Implementation Details

Here we provide some more details about the exact implementation of our experiments.

Multi-MNIST and CLEVR

To train our GAN approach on the Multi-MNIST (CLEVR) data set we use the Stage-I Generator and Discriminator from the StackGAN MS-COCO architecture[1]. In our following description an **upsample block** describes the following sequence: nearest neighbor upsampling with factor 2, a convolutional layer with X filters (filter size 3×3, stride 1, padding 1), batch normalization, and a ReLU activation. The bounding box labels are one-hot vectors of size $[1, 10]$ encoding the digit identity (CLEVR: $[1, 13]$ encoding object shape and color). Please refer to Table B.2 for detailed information on the individual layers described in the following. For all leaky ReLU activations *alpha* was set to 0.2.

In the **object pathway of the generator** we first create a zero tensor $\mathbb{O}_G$ which will contain the feature representations of the individual objects. We then spatially replicate each bounding box label into a 4×4 layout of shape $(10, 4, 4)$ (CLEVR: $(13, 4, 4)$) and apply two upsampling blocks. The resulting tensor is then added to the tensor $\mathbb{O}_G$ at the location of the bounding box using a spatial transformer network.

In the **global pathway of the generator** we first obtain the layout encoding. For this we create a tensor of shape $(10, 16, 16)$ (CLEVR: $(13, 16, 16)$) that contains the one-hot labels at the location of the bounding boxes and is zero everywhere else. We then apply three convolutional layers, each followed by batch normalization and a leaky ReLU activation. We reshape the output to shape $(1, 64)$ and concatenate it with the noise tensor of shape $(1, 100)$ (sampled from a random normal distribution) to form a tensor of shape $(1, 164)$. This tensor is then fed into a dense layer, followed by batch normalization and a ReLU activation and the output is reshaped to $(-1, 4, 4)$. We then apply two upsampling blocks to obtain a tensor of shape $(-1, 16, 16)$.

At this point, the outputs of the object and the global pathway are concatenated along the channel axis to form a tensor of shape $(-1, 16, 16)$. We then apply another two upsampling blocks resulting in a tensor of shape $(-1, 64, 64)$ followed by a convolutional layer and a TanH activation to obtain the final image of shape $(-1, 64, 64)$.

In the **object pathway of the discriminator** we first create a zero tensor $\mathbb{O}_D$ which will contain the feature representations of the individual objects. We then use a spatial transfomer network to extract the image features at the locations of the bounding boxes and reshape them to a tensor of shape $(1, 16, 16)$ (CLEVR: $(3, 16, 16)$). The one-hot label of each bounding box are spatially replicated to a shape of $(10, 16, 16)$ (CLEVR: $(13, 16, 16)$) and concatenated with the previously extracted features to form a tensor of shape $(11, 16, 16)$ (CLEVR: $(16, 16, 16)$). We then apply a convolutional layer, batch normalization and a leaky ReLU activation to the concatenation of features and label and, again, use a spatial transformer network to resize the output to the shape of the respective bounding box before adding it to the tensor $\mathbb{O}_D$.

In the **global pathway of the discriminator**, we apply two convolutional

[1] https://github.com/hanzhanggit/StackGAN-Pytorch

layers, each followed by batch normalization and a leaky ReLU activation and concatenate the resulting tensor with the output of the object pathway. After this, we again apply two convolutional layers, each followed by batch normalization and a leaky ReLU activation. We concatenate the resulting tensor with the conditioning information about the image content, in this case, the sum of all one-hot vectors. To this tensor we apply another convolutional layer, batch normalization, a leaky ReLU activation, and another convolutional layer, to obtain the final output of the discriminator of shape (1).

Similarly to the procedure of StackGAN and other conditional GANs we train the discriminator to classify real images with correct labels (the sum of one-hot vectors supplied in the last step of the process) as real, while generated images with correct labels and real images with (randomly sampled) incorrect labels should be classified as fake.

MS-COCO

StackGAN-Stage-I For training the Stage-I generator and discriminator (images of size 64×64 pixels) we follow the same procedure and architecture outlined in the previous section about the training on the Multi-MNIST and CLEVR data sets. The only difference is that we now have **image captions** as an additional description of the image. As such, to obtain the bounding box labels we concatenate the image caption embedding[2] and the one-hot encoded bounding box label and apply a dense layer with 128 units, batch normalization, and a ReLU activation to it, to obtain a label of shape $(1, 128)$ for each bounding box. In the final step of the discriminator when we concatenate the feature representation with the conditioning vector, we use the image encoding as conditioning vector and do not use any bounding box labels at this step. The rest of the training proceeds as described in the previous section, except that the bounding box labels now have a shape of $(1, 128)$. All other details can be found in Table B.2.

StackGAN-Stage-II In the second part of the training, we train a second generator and discriminator to generate images with a resolution of 256×256 pixels. The generator gets as input images with a resolution of 64×64 pixels (generated by the trained Stage-I generator) and the image caption and uses them to generate images with a 256×256 pixels resolution. A new discriminator is trained to distinguish between real and generated images.

On the **Stage-II generator** we perform the following modifications we use the same procedure as in the Stage-I generator to obtain the **bounding box labels**. To obtain an **image encoding** from the generated 64×64 image we use three convolutional layers, each followed by batch normalization and a ReLU activation to obtain a feature representation of shape $[-1, 16, 16]$. Additionally, we replicate each bounding box label (obtained with the dense layer) spatially at the locations of the bounding boxes on an empty canvas of shape $[128, 16, 16]$ and then concatenate

[2]Downloaded from https://github.com/reedscot/icml2016

it along the channel axis with the image encoding and the spatially replicated image caption embedding. As in the standard StackGAN we then apply more convolutional layers with residual connections to obtain the final image embedding of shape $[-1, 16, 16]$, which provides the input for both the object and the global pathway.

The **generator's object pathway** gets as input the image encoding described in the previous step. First, we create a zero tensor $\mathbb{O}_G$ which will contain the feature representations of the individual objects. We then use a spatial transformer network to extract the features from within the bounding box and reshapes those features to $[-1, 16, 16]$. After this, we apply two upsample blocks and then use a spatial transformer network to add the features to $\mathbb{O}_G$ within the bounding box region. This is done for each of the bounding boxes within the image.

The **generator's global pathway** gets as input the image encoding and uses the same convolutional layers and upsampling procedures as the original StackGAN Stage-II generator. The **outputs of the object and global pathway** are merged at the resolution of $[-1, 64, 64]$ by concatenating the two outputs along the channel axis. After this, we continue using the standard StackGAN architecture to generate images of shape $[3, 256, 256]$.

The **Stage-II discriminator's object pathway** first creates a zero tensor $\mathbb{O}_D$ which will contain the feature representations of the individual objects. It gets as input the image (resolution of 256×256 pixels) and we use a spatial transformer network to extract the features from the bounding box and reshape those features to a shape of $[3, 32, 32]$. We spatially replicate the bounding box label (one-hot encoding) to a shape of $[-1, 32, 32]$ and concatenate it with the extracted features along the channel axis. This is then given to the object pathway which consists of two convolutional layers with batch normalization and a LeakyReLU activation. The output of the object pathway is again transformed to the width and height of the bounding box with a spatial transformer network and then added to $\mathbb{O}_D$. This procedure is performed with each of the bounding boxes within the image (maximum of three during training).

The **Stage-II discriminator's global pathway** consists of the standard StackGAN layers, i.e. it gets as input the image (256×256 pixels) and applies convolutional layers with stride 2 to it. The **outputs of the object and global pathways** are merged at the resolution of $[-1, 32, 32]$ by concatenating the two outputs along the channel axis We then apply more convolutional with stride 2 to decrease the resolution. After this, we continue in the same way as the original StackGAN.

AttnGAN On the AttnGAN[3] we only modify the training at the lower layers of the generator and the first discriminator (working on images of 64×64 pixels resolution). For this, we perform the same modifications as described in the StackGAN-Stage-I generator and discriminator. In the **generator** we obtain the bounding box labels in the same way as in the StackGAN, by concatenating

[3] https://github.com/taoxugit/AttnGAN

the image caption embedding with the respective one-hot vector and applying a dense layer with 100 units, batch normalization, and a ReLU activation to obtain a bounding box label. In contrast to the previous architectures, we follow the AttnGAN implementation in use the gated linear unit function (GLU) as standard activation for our convolutional layers in the generator.

In the **generator's object pathway** we first create a zero tensor $\mathbb{O}_G$ of shape $(192, 16, 16)$ which will contain the feature representations of the individual objects. We then spatially replicate each bounding box label into a 4×4 layout of shape $(100, 4, 4)$ and apply two upsampling blocks with 768 and 384 filters (filter size=3, stride=1, padding=1). The resulting tensor is then added to the tensor $\mathbb{O}_G$ at the location of the bounding box using a spatial transformer network.

In the **global pathway of the generator** we first obtain the layout encoding in the same way as in the StackGAN-I generator, except that the three convolutional layers of the layout encoding now have 50, 25, and 12 filters respectively (filter size=3, stride=2, padding=1). We concatenate it with the noise tensor of shape $(1, 100)$ (sampled from a random normal distribution) and the image caption embedding to form a tensor of shape $(1, 248)$. This tensor is then fed into a dense layer with 24,576 units, followed by batch normalization and a ReLU activation and the output is reshaped to $(768, 4, 4)$. We then apply two upsampling blocks with 768 and 384 filters to obtain a tensor of shape $(192, 16, 16)$.

At this point the **outputs of the object and the global pathways** are concatenated along the channel axis to form a tensor of shape $(384, 16, 16)$. We then apply another two upsampling blocks with 192 and 96 filters, resulting in a tensor of shape $(48, 64, 64)$. This feature representation is then used by the following layers of the AttnGAN generator in the same way as detailed in the original paper and implementation.

In the **object pathway of the discriminator** we first create a zero tensor $\mathbb{O}_D$ which will contain the feature representations of the individual objects. We then use a spatial transfomer network to extract the image features at the locations of the bounding boxes and reshape them to a tensor of shape $(3, 16, 16)$. The one-hot label of each bounding box is spatially replicated to a shape of $(-1, 16, 16)$ and concatenated with the previously extracted features. We then apply a convolutional layer with 192 filters (filter size=4, stride=1, padding=1), batch normalization and a leaky ReLU activation to the concatenation of features and label and, again, use a spatial transformer network to resize the output to the shape of the respective bounding box before adding it to the tensor $\mathbb{O}_D$.

In the **global pathway of the discriminator** we apply two convolutional layers with 96 and 192 filters (filter size=4, stride=2, padding=1), each followed by batch normalization and a leaky ReLU activation and concatenate the resulting tensor with the output of the object pathway. After this, we again apply two convolutional layers with 384 and 768 filters (filter size=4, stride=2, padding=1), each followed by batch normalization and a leaky ReLU activation. We concatenate the resulting tensor with the spatially replicated image caption embedding. To this tensor we apply another convolutional layer with 768 filters (filter size=3, stride=1, padding=1), batch normalization, a leaky ReLU activation, and another

convolutional layer with one filter (filter size=4, stride=4, padding=0), to obtain the final output of the discriminator of shape (1). The rest of the training and all other hyperparameters and architectural values are left the same as in the original implementation.

	Multi-MNIST	CLEVR	MS-COCO-I	MS-COCO-II
Optimizer	Adam ($beta_1 = 0.5$, $beta_2 = 0.999$)			
Learning Rate	0.0002	0.0002	0.0002	0.0002
Schedule: halve every x epochs	10	20	20	20
Training Epochs	20	40	120	110
Batch Size	128	128	128	40
Weight Initialization	$\mathcal{N}(0, 0.02)$	$\mathcal{N}(0, 0.02)$	$\mathcal{N}(0, 0.02)$	$\mathcal{N}(0, 0.02)$
Z-Dim / Img-Caption-Dim	100 / 10	100 / 13	100 / 128	100 / 128
Generator				
Image Encoder				
Conv (fs=3, s=1, p=1)				192
Conv (fs=4, s=2, p=1)				384
Conv (fs=4, s=2, p=1)				768
Concat with image caption and bbox labels				$(1024, 16, 16)$
Conv (fs=3, str=1, pad=1)				768
$4 \times$ Res. (fs=3, s=1, p=1)				768
Object Pathway				
$\mathbb{O}_G$ Shape	$(256, 16, 16)$	$(192, 16, 16)$	$(384, 16, 16)$	$(192, 64, 64)$
Upsample (fs=3, s=1, p=1)	512	384	768	384
Upsample (fs=3, s=1, p=1)	256	192	384	192
Output Shape	$(256, 16, 16)$	$(192, 16, 16)$	$(384, 16, 16)$	$(192, 64, 64)$
Global Pathway				
Layout Encoding				
Conv (fs=3, s=2, p=1)	64	64	64	
Conv (fs=3, s=2, p=1)	32	32	32	
Conv (fs=3, s=2, p=1)	16	16	16	
Dense Layer Units	16,384	12,288	24,576	
Upsample (fs=3, s=1, p=1)	512	384	768	384
Upsample (fs=3, s=1, p=1)	256	192	384	192
Output Shape	$(256, 16, 16)$	$192, 16, 16)$	$(384, 16, 16)$	$(192, 64, 64)$
Concat outputs of object and global pathways	$(512, 16, 16)$	$(384, 16, 16)$	$(768, 16, 16)$	$(384, 64, 64)$
Upsample (fs=3, s=1, p=1)	128	96	192	96
Upsample (fs=3, s=1, p=1)	64	48	96	48
Conv (fs=3, s=1, p=1)	1	3	3	3
Generator Output	$(1, 64, 64)$	$(3, 64, 64)$	$(3, 64, 64)$	$(3, 256, 256)$
Discriminator				
Object Pathway				
$\mathbb{O}_D$ Shape	$(128, 16, 16)$	$(96, 16, 16)$	$(192, 16, 16)$	$(192, 32, 32)$
Conv (fs=4, s=1, p=1)	128	96	192	192
Conv (fs=4, s=1, p=1)				192
Output Shape	$(128, 16, 16)$	$(96, 16, 16)$	$(192, 16, 16)$	$(192, 32, 32)$
Global Pathway				
Conv (fs=4, s=2, p=1)	64	48	96	96
Conv (fs=4, s=2, p=1)	128	96	192	192
Conv (fs=4, s=2, p=1)				384
Output Shape	$(128, 16, 16)$	$(96, 16, 16)$	$(192, 16, 16)$	$(384, 32, 32)$
Concat outputs of object and global pathways	$(256, 16, 16)$	$(192, 16, 16)$	$(384, 16, 16)$	$(576, 32, 32)$
Conv (fs=4, s=2, p=1)	256	192	384	768
Conv (fs=4, s=2, p=1)	512	384	768	1,536
Conv (fs=4, s=2, p=1)				3,072
Conv (fs=3, s=1, p=1)				1,536
Conv (fs=3, s=1, p=1)				768
Concat with conditioning vector	$(522, 4, 4)$	$(397, 4, 4)$	$(896, 4, 4)$	$(896, 4, 4)$
Conv (fs=3, s=1, p=1)	512	384	768	768
Conv (fs=4, s=4, p=0)	1	1	1	1

Table B.2: Overview of the individual layers used in our networks to generate images of resolution 64×64 / 256×256 pixels. Values in brackets (C, H, W) represent the tensor's shape. Numbers in the columns after convolutional, residual, or dense layers describe the number of filters / units in that layer. (fs=x, s=y, p=z) describes filter size, stride, and padding for that convolutional / residual layer.

B.2.2 Additional Examples of Multi-MNIST Results: Training and test set over complementary regions

Figure B.1: Systematic test of digits over vertically different regions. Training set included three normal-sized digits only in the top half of the image. Highlighted bounding boxes and yellow ground truth for visualization. We can see that the model fails to generate recognizable digits once their location is too far in the bottom half of the image, as this location was never observed during training.

B.2.3 Additional Examples of MS-COCO Results: StackGAN

Figure B.2 shows results of text-to-image synthesis on the MS-COCO data set with the StackGAN architecture. Rows A show the original image and image caption, rows B show the images generated by our StackGAN + Object Pathway and the given bounding boxes for visualization, and rows C show images generated by the original StackGAN (pretrained model obtained from https://github.com/hanzhanggit/StackGAN-Pytorch). The last block of examples (last row) show typical failure cases of our model, where there is no bounding box for the foreground object present. As a result our model only generates the background, without the appropriate foreground object, even though the foreground object is very clearly described in the image caption. Figure B.4 provides similar results but for random bounding box positions. The first six examples show images generated by our StackGAN where we changed the location and size of the respective bounding boxes. The last three examples show failure cases in which we changed the location of the bounding boxes to "unusual" locations. For the image with the child on the bike, we put the bounding box of the bike somewhere in the top half of the image and the bounding box for the child somewhere in the bottom part. Similarly, for the man sitting on a bench, we put the bench in the top and the man in the bottom

half of the image. Finally, for the image depicting a pizza on a plate, we put the plate location in the top half of the image and the pizza in the bottom half.

B.2.4 Additional Examples of MS-COCO Results: AttnGAN

Figure B.3 shows results of text-to-image synthesis on the MS-COCO data set with the AttnGAN architecture. Rows A show the original image and image caption, rows B show the images generated by our AttnGAN + Object Pathway and the given bounding boxes for visualization, and rows C show images generated by the original AttnGAN (pretrained model obtained from https://github.com/taoxugit/AttnGAN). The last block of examples (last row) show typical failure cases, in which the model does generate the appropriate object within the bounding box, but also places the same object at multiple other locations within the image. Similarly as for StackGAN, Figure B.5 shows images generated by our AttnGAN where we randomly change the location of the various bounding boxes. Again, the last three examples show failure cases where we put the locations of the bounding boxes at "uncommon" positions. In the image depicting the sandwiches we put the location of the plate in the top half of the image, in the image with the dogs we put the dogs' location in the top half, and in the image with the motorbike we put the human in the left half and the motorbike in the right half of the image.

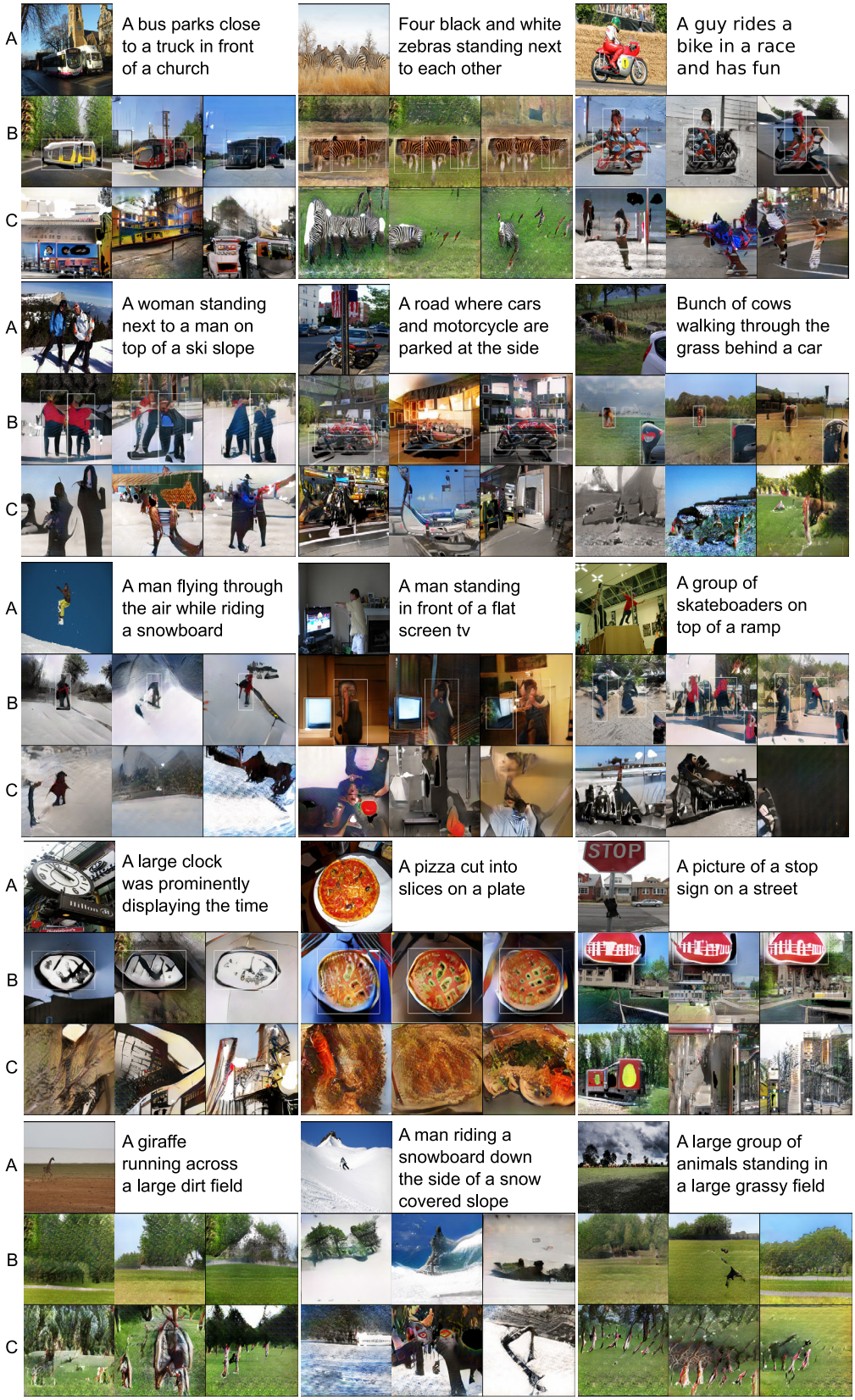


Figure B.2: Additional StackGAN examples.

Figure B.3: Additional AttnGAN examples.

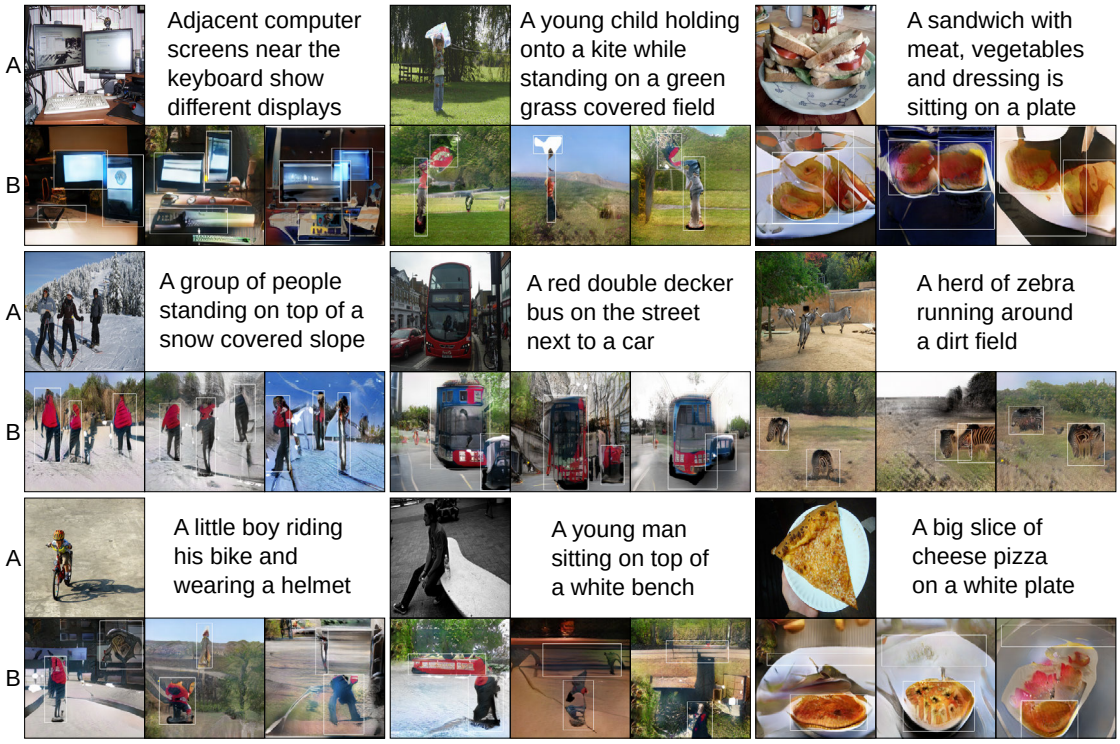


Figure B.4: StackGAN examples with random locations.

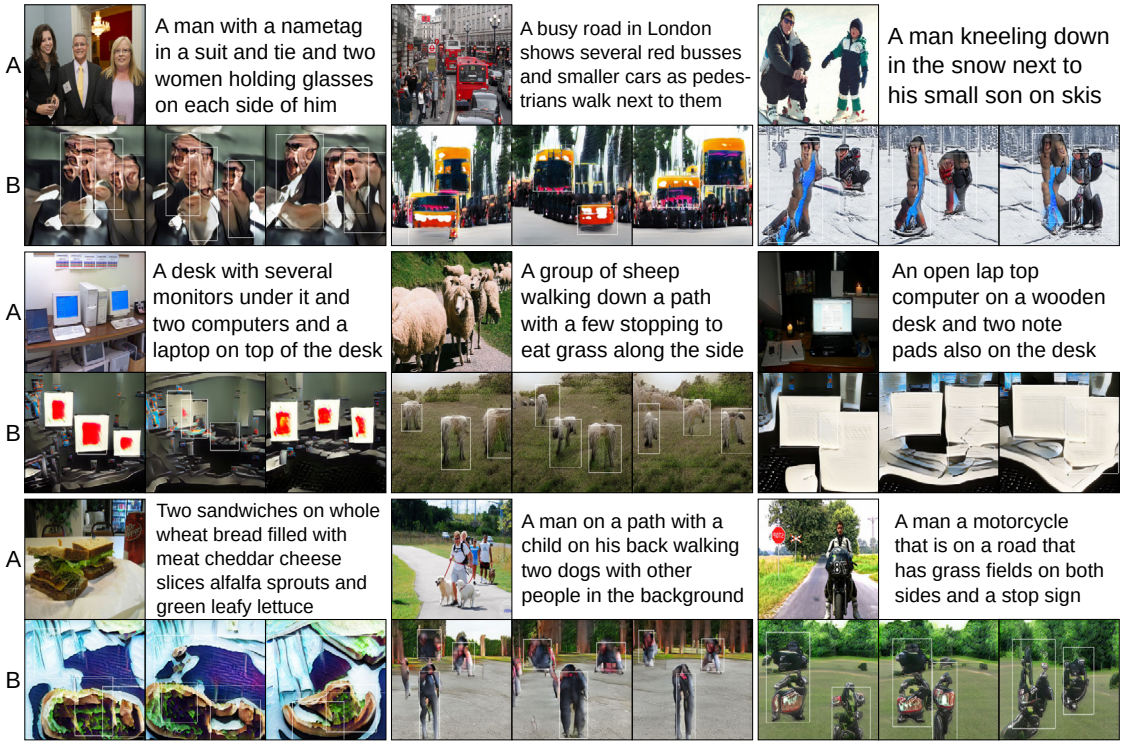


Figure B.5: AttnGAN examples with random locations.

B.2.5 Object Detection on MS-COCO Images

To further inspect the quality and recognizability of the generated objects, we ran a test on object detection using a YOLOv3 network Redmon and Farhadi [2018] that was pretrained on the MS-COCO data set[4]. We use the Pytorch implementation from https://github.com/ayooshkathuria/pytorch-yolo-v3 to get the bounding box and label predictions. We follow the standard guidelines and keep all hyperparameters for the YOLOv3 network as in the implementation. We picked the 30 most common training labels (based on how many captions contain these labels) and evaluate the models on these labels, see Table B.3.

In the following, we evaluate how often the pretrained YOLOv3 network recognizes a specific object within a generated image that should contain this object based on the image caption. For example, we expect an image generated from the caption *"a young woman taking a picture with her phone"* to contain a person somewhere in the image and we check whether the YOLOv3 network actually recognizes a person in the generated image. Since the baseline StackGAN and AttnGAN only receive the image caption as input (no bounding boxes and no bounding box labels) we decided to only use captions that clearly imply the presence of the given label (see Table B.3). We chose this strategy in order to allow for a fair comparison of the resulting presence or absence of a given object. Specifically, for a given label we choose all image captions from the test set that contain one of the associated words for this label (associated words were chosen manually, see Table B.3) and then generated three images for each caption with each model. Finally, we counted the number of images in which the given object was detected by the YOLOv3 network. Table B.4 shows the ratio of images for each label and each model in which the given object was detected at any location in the image.

Additionally, for our models that also receive the bounding boxes as input, we calculated the Intersection over Union (IoU) between the ground truth bounding box (the bounding box supplied to the model) and the bounding box predicted by the YOLOv3 network for the recognized object. Table B.4 presents the average IoU (for the models that have an object pathway) for each object in the images in which YOLOv3 detected the given object. For each image in which YOLOv3 detected the given object, we calculated the IoU between the predicted bounding box and the ground truth bounding box for the given object. In the cases in which either an image contains multiple instances of the given object (i.e. multiple different bounding boxes for this object were given to the generator) or YOLOv3 detects the given object multiple times we used the maximum IoU between all predicted and ground truth bounding boxes for our statistics.

Figure B.6 visualizes how the IoU and recall values are distributed for the different models, and Table B.4 summarizes the results with the 30 tested labels. We can observe that the StackGAN with object pathway outperforms the original StackGAN when comparing the recall of the YOLOv3 network, i.e. in how many images with a given label the YOLOv3 network actually detected the given object. The recall of the original StackGAN is higher than 10% for 26.7% of the labels,

[4]Pretrained weights from the author, acquired via: https://pjreddie.com/darknet/yolo/

Label	Occurrences	Words in captions	Label	Occurrences	Words in captions
Person	13773	person, people, human, man, men, woman, women, child	Umbrella	727	umbrella
			Elephant	708	elephant
			Chair	632	chair, stool
Dining table	3130	table, desk	Zebra	627	zebra
Car	1694	car, auto, vehicle, cab	Boat	627	boat
Cat	1658	cat	Bird	610	bird
Dog	1543	dog	Aeroplane	602	plane
Bus	1198	bus	Bicycle	600	bicycle
Train	1188	train	Surfboard	595	surfboard
Bed	984	bed	Kite	593	kite
Pizza	906	pizza	Truck	561	truck
Horse	874	horse	Stop sign	522	stop
Giraffe	828	giraffe	TV Monitor	471	tv, monitor, screen
Toilet	797	toilet	Sofa	467	sofa, couch
Bear	777	bear	Sandwich	387	sandwich
Bench	732	bench	Sheep	368	sheep

Table B.3: Words that were used to identify given labels in the image caption for the YOLOv3 object detection test.

while our StackGAN with object pathway results in a recall greater than 10% for 60% of the labels. The IoU is greater than 0.3 for every label, while 86.7% of the labels result an IoU of greater than 0.5 (original images: 100%) and 30% have an IoU of greater than 0.7 (original images: 96.7%). This indicates that we can indeed control the location and identity of various objects within the generated images.

Compared to the StackGAN, the AttnGAN achieves a much greater recall, with 80% and 83.3% of the labels having a recall of greater than 10% for the original AttnGAN and the AttnGAN with object pathway respectively. The difference in recall values between the original AttnGAN and the AttnGAN with object pathway is also smaller, with our AttnGAN having a higher (lower) recall than the original AttnGAN (we only count cases where the difference is at least 5%) in 26.7% (13.3%) of the labels. The average IoU, on the other hand, is a lot smaller for the AttnGAN than for the StackGAN. We only achieve an IoU greater than 0.3 (0.5, 0.7) for 53.3% (3.3%, 0%) of the labels. We attribute this to the observation that the AttnGAN tends to place seemingly recognizable features of salient objects at arbitrary locations throughout the image. This might attribute to the overall higher recall but may negatively affect the IoU.

Overall, these results further confirm our previous experiments and highlight that the addition of the object pathway to the different models does not only enable the direct control of object location and identity but can also help to increase the image quality. The increase in image quality is supported by a higher Inception Score, lower Fréchet Inception Distance (for StackGAN) and a higher performance of the YOLOv3 network in detecting objects within generated images.

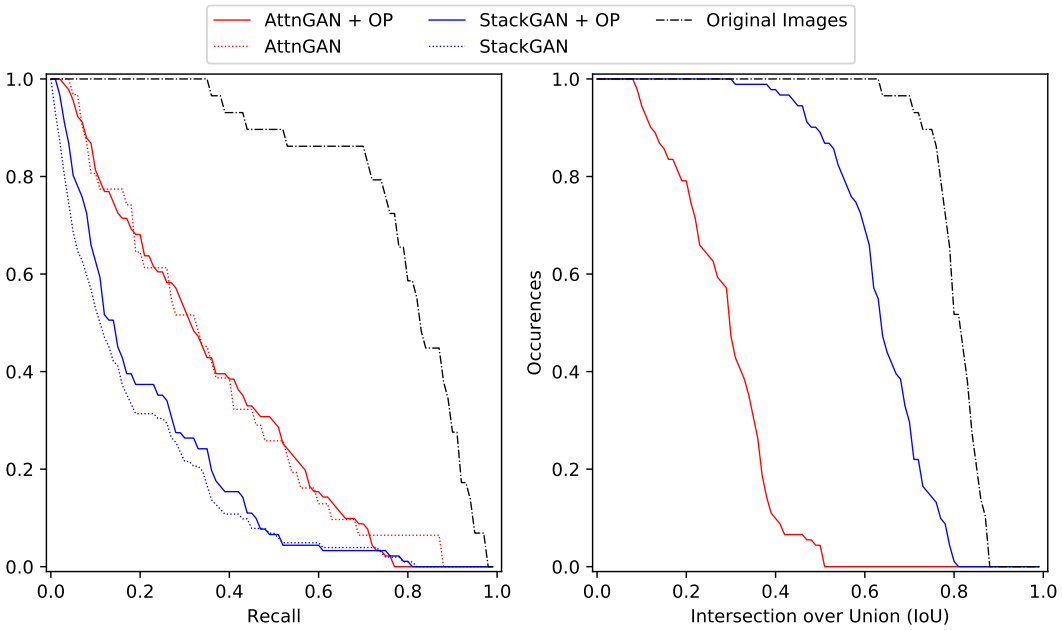


Figure B.6: Recall and IoU values in the YOLOv3 object detection test.

B.3 Evaluating Compositional Representations

This section gives more details about our experiments outlined in Section 4.3 and also shows additional results.

B.3.1 Information about Captions for SOA

Table B.5 gives a detailed overview of how we chose the captions for each label to calculate the *Semantic Object Accuracy* (SOA) scores. The second column shows how many captions we found in total for the given label. The third column shows which words we filtered the captions for to obtain captions for the given label. This means that we chose all captions that contained at least one of those words as a valid caption for the given label. In the fourth column we show (were applicable) which words were explicitly excluded when looking for captions for the given label. Finally, the last column shows some examples of "false positives", i.e. captions that are included in the set of captions for the given label even though they do not necessarily explicitly ask for the presence of the given label as understood by humans.

B.3.2 Inspection of YOLO Predictions

Figure B.7 shows generated images with the ground truth bounding boxes (red) provided as input to the model and the bounding boxes detected by YOLO (blue). When the Intersection over Union (IoU) is small (right column) we observe that this is usually due to the fact that the generated object is much larger than the

Label	Orig. Img.		StackGAN	StackGAN + OP		AttnGAN	AttnGAN + OP	
	Recall	IoU	Recall	Recall	IoU	Recall	Recall	IoU
Person	.943	.824	.355	.451 ± .019	.624 ± .012	.598	.610 ± .008	.276 ± .006
Dining table	.355	.774	.007	.022 ± .004	.734 ± .011	.069	.045 ± .022	.490 ± .018
Car	.433	.792	.012	.047 ± .007	.622 ± .020	.006	.063 ± .010	.144 ± .043
Cat	.715	.821	.021	.104 ± .100	.622 ± .008	.423	.430 ± .066	.350 ± .012
Dog	.703	.819	.068	.150 ± .007	.601 ± .004	.450	.488 ± .048	.311 ± .007
Bus	.747	.877	.161	.393 ± .031	.794 ± .009	.352	.416 ± .032	.374 ± .006
Train	.900	.835	.133	.310 ± .033	.700 ± .007	.393	.438 ± .110	.355 ± .036
Bed	.775	.789	.032	.141 ± .018	.701 ± .001	.539	.552 ± .030	.505 ± .002
Pizza	.912	.842	.119	.485 ± .101	.786 ± .004	.444	.660 ± .054	.395 ± .016
Horse	.933	.842	.129	.330 ± .048	.585 ± .039	.532	.619 ± .027	.300 ± .006
Giraffe	.972	.857	.173	.467 ± .035	.606 ± .030	.472	.650 ± .084	.365 ± .030
Toilet	.898	.826	.005	.122 ± .021	.690 ± .010	.201	.220 ± .021	.224 ± .011
Bear	.381	.859	.015	.120 ± .018	.720 ± .036	.319	.303 ± .028	.357 ± .010
Bench	.828	.798	.001	.030 ± .008	.627 ± .034	.094	.094 ± .031	.308 ± .018
Umbrella	.912	.762	.001	.023 ± .009	.578 ± .030	.060	.063 ± .017	.154 ± .053
Elephant	.940	.867	.060	.414 ± .069	.688 ± .033	.350	.500 ± .141	.353 ± .006
Chair	.757	.755	.014	.039 ± .004	.488 ± .039	.070	.093 ± .005	.225 ± .001
Zebra	.972	.875	.732	.781 ± .023	.686 ± .017	.870	.766 ± .063	.315 ± .022
Boat	.795	.709	.077	.010 ± .011	.594 ± .021	.168	.202 ± .027	.206 ± .020
Bird	.837	.781	.059	.097 ± .027	.500 ± .066	.322	.357 ± .042	.250 ± .020
Aeroplane	.912	.812	.125	.223 ± .043	.667 ± .026	.499	.415 ± .010	.320 ± .035
Bicycle	.825	.760	.007	.053 ± .020	.558 ± .052	.170	.191 ± .013	.233 ± .024
Surfboard	.873	.780	.030	.067 ± .019	.459 ± .056	.104	.110 ± .025	.143 ± .016
Kite	.772	.633	.029	.057 ± .028	.426 ± .086	.260	.162 ± .068	.120 ± .018
Truck	.887	.832	.082	.243 ± .062	.717 ± .022	.378	.367 ± .027	.393 ± .019
Stop Sign	.527	.874	.001	.261 ± .057	.780 ± .011	.070	.124 ± .048	.101 ± .014
TV Monitor	.818	.833	.037	.264 ± .005	.765 ± .016	.529	.435 ± .314	.243 ± .066
Sofa	.878	.794	.012	.087 ± .024	.628 ± .044	.170	.191 ± .057	.329 ± .028
Sandwich	.792	.796	.045	.139 ± .049	.628 ± .014	.340	.370 ± .054	.318 ± .031
Sheep	.943	.727	.004	.091 ± .006	.460 ± .011	.250	.304 ± .037	.116 ± .022

Table B.4: Results of YOLOv3 detections on generated and original images. Recall provides the fraction of images in which YOLOv3 detected the given object. *IoU* (Intersection over Union) measures the maximum IoU per image in which the given object was detected.

originally provided bounding box. This agrees with our hypothesis that the reason for the relatively small IoU numbers for our model is because it tends to put salient object features even at locations outside of the provided bounding box. Note that our model rarely generates the desired object at a location completely different from the provided bounding box. Rather, it tends to increase the object's size, especially when the provided bounding box is small. However, we can also see that most objects are not clearly recognizable to humans even though they are "correctly" detected by the YOLO network. This is in line with our observation that YOLO, like many other CNNs, tend to be focused on textural cues much more than on shapes. As a result, future improvements in object detection models can also help increase the information provided by our SOA score.

B.3.3 Model Architecture

Table B.6 and Table B.6 show our model's architecture. We train our model on four NVIDIA GeForce GTX 1080Ti GPUs. Training one model takes between two and four weeks, depending on the exact setting.

B.3.4 Further Results

Table B.8 and Table B.9 show the detailed results of the YOLOv3 detection network on the individual labels for all models.

Table B.5: Words that were used to identify given labels in the image caption for the YOLOv3 object detection test (plural of each word also included, different forms of spelling also included).

Label	# Sent.	Words in Captions	Excluded Strings	False Positives
Person	61586	person, people, human, man, men, woman, women, child, children		A sign advertising an eatery in which people can eat burgers.
Dining Table	7678	table, desk		A sweet dish is kept in a bowl on a table mat.
Cat	6609	cat, kitten		A double parking meter decorated with cat art
Dog	5614	dog, pup	hot dog, hotdog, hot-dog, cheese dog, chili dog, corn dog	Two stuffed dogs under a blanket looking at a picture book.
Train	5397	train		A red train engine sits on the tracks
Bus	4027	bus		The sign is pointing the direction of the bus route.
Clock	3870	clock		
Giraffe	3866	giraffe		A woman standing in front of a giraffe pen
Pizza	3655	pizza		Would you prefer fresh basil on your pizza or sans basil?
Horse	3615	horse		A close-up of a man hating the horses face.
Elephant	3133	elephant	toy elephant, stuffed elephant	Outdoor art display of elephant sculptures of various colorings.
Zebra	3070	zebra		An animal that is part horse and part zebra by another horse.
Bed	2923	bed		A large truck has a flat bed trailer attached
Boat	2819	boat, ship		a upside down boat is on top of a big hil
Toilet	2796	toilet		You can pick either toilet stall in this clean restroom.
Bird	2691	bird		a clock with a painting of a bird on a branch on it
Skateboard	2665	skateboard		
Car	2650	car, auto	train car, car window, side car, passenger car, subway car, car tire, rail car, tram car, street car, trolly car	A museum sign showing the main entrance and car park
Bench	2633	bench		
Laptop	2376	laptop		
Surfboard	2270	surfboard		
Truck	2213	truck		
Umbrella	2107	umbrella		a man playing with a white ball on a red umbrella
Kite	2025	kite	kite board, kiteboard	
Sports Ball	2001	ball		Female tennis player looks on as she waits for the ball serve
Cake	2012	cake	cupcake	
Cow	1981	cow		A young boy sitting on top of a cow statue.
Bicycle	1920	bike, bicycle	motorbike, motor bike, motorcycle, dirt bike	A man drives his bike taxi with luggage in the back.
Chair	1884	chair		
Frisbee	1775	frisbee		
Bear	1740	bear	teddy bear, stuffed bear, care bear, toy bear	a very old panda bear doll with a handkerchief

Sandwich	1649	sandwich		
Sheep	1626	sheep		furniture shaped like sheep on a open field
Vase	1597	vase		
Bowl	1570	bowl	toilet bowl	
Sink	1529	sink		
Stop Sign	1491	stop sign		That sign almost looks like a stop sign with no words on it.
Banana	1466	banana		
Monitor	1437	monitor, tv, screen		Four cell phone on a wooden table with their screens on.
Skis	1419	skis		
Hot Dog	1717	hot dog, chili dog, cheese dog, corn dog		
Fire Hydrant	1408	hydrant		
Sofa	1404	sofa, couch		
Teddybear	1284	teddybear		
Aeroplane	1195	plane, jet, aircraft		Mountaineous view as seen from a jet airliner
Tie	1062	tie	to tie	
Tennis Racket	993	racket		
Cell Phone	956	cell phone, mobile phone		
Refrigerator	949	refrigerator, fridge		
Cup	902	cup		A table with measuring cups and bowls on it
Broccoli	840	broccoli		
Donut	805	donut		
Bottle	766	bottle		A toy hot dog and ketchup bottle on a table
Suitcase	736	suitcase		
Snowboard	732	snowboard		
Book	731	book		A large open room has an overhead book shelf
Remote	670	remote		
Traffic Light	645	traffic light		
Keyboard	603	keyboard		
Apple	510	apple	pineapple	
Oven	506	oven	microwave oven	
Motorcycle	495	motorcycle, dirt bike, motorbike, scooter		A group of dirt bike racers in a row
Carrot	463	carrot		
Scissor	450	scissors		
Parking Meter	430	parking meter		
Microwave	416	microwave		
Orange	378	oranges		
Knife	376	knife		
Fork	363	fork		A large fork sculpture stands in the water as a large boat passes
Baseball Bat	322	baseball bat		
Toothbrush	267	toothbrush		
Wine Glass	264	wine glass		
Backpack	220	backpack, rucksack		
Spoon	206	spoon		
Handbag	107	handbag, purse		Items from a handbag laid out neatly on a carpet
Toaster	89	toaster		
Potted Plant	81	potted plant		
Mouse	72	computer mouse		
Baseball Glove	39	baseball glove		
Hair Drier	35	hair drier		

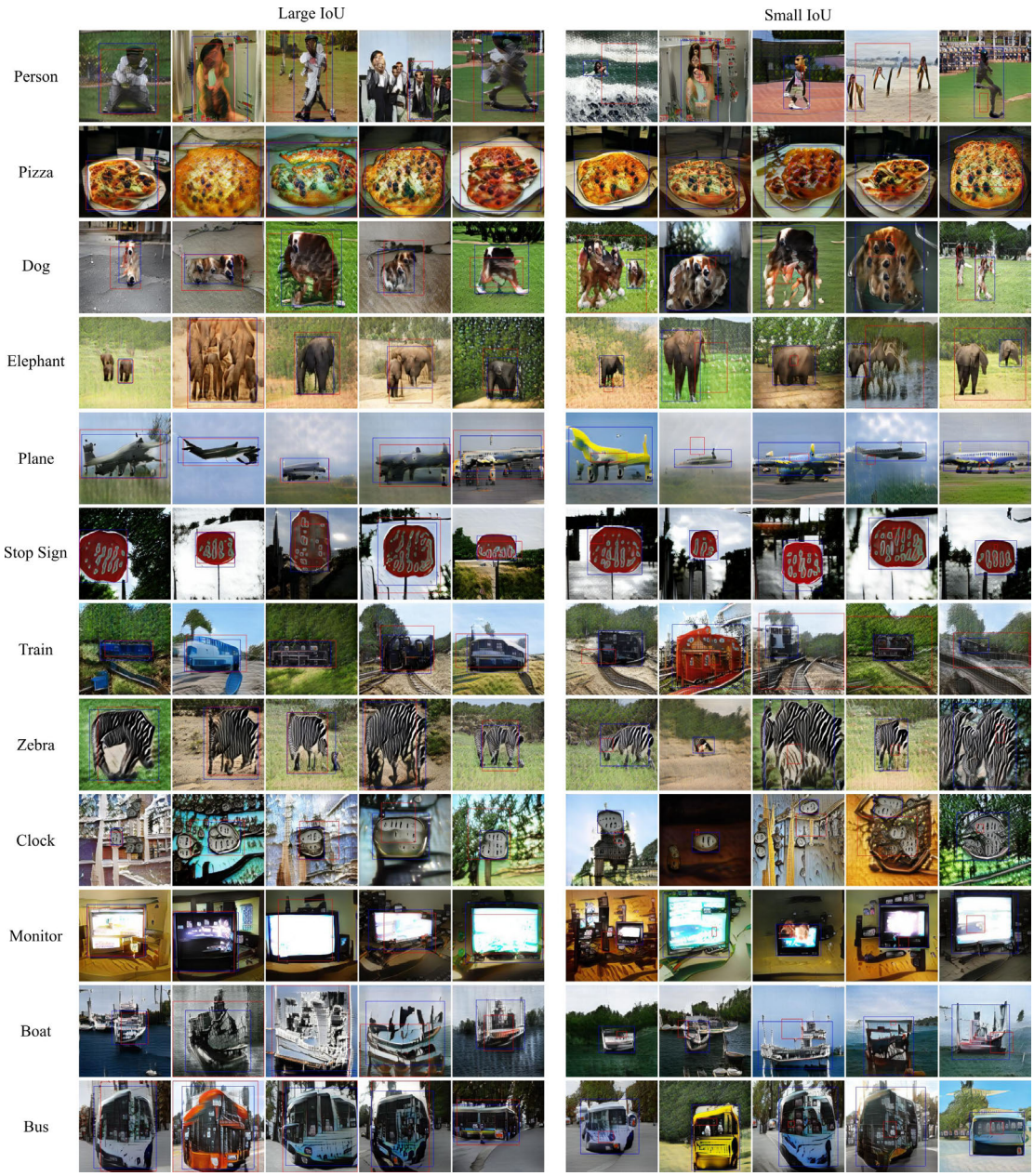


Figure B.7: Examples of our model and YOLOv3 predictions on the generated images. The bounding boxes in red are the bounding boxes provided to the network at test time for the given objects. The blue bounding boxes are the bounding boxes provided by YOLOv3 for the given object. When the Intersection over Union (IoU) is small (right column) we observe that this is usually due to the fact that the generated object is much larger than the originally provided bounding box. Only in few cases is the generated object at a completely different location than the provided red bounding box.

Table B.6: Overview of the individual layers used in our networks to generate images of resolution 256×256 pixels. Values in brackets (C, H, W) represent the tensor's shape. Numbers in the columns after convolutional, residual, or dense layers describe the number of filters / units in that layer. (fs=x, s=y, p=z, BN=B) describes the filter size, stride, padding, and batch norm for that convolutional / residual layer. Everything not specifically mentioned or explained (e.g. RNN-Encoder, DAMSM) is the same as in the AttnGAN [Xu et al., 2018b].

Optimizer: Adam	($\beta_1 = 0.5$, $\beta_2 = 0.999$)
Activation Functions	Relu (RL), Leaky RL (LR), Gated Linear Unit (GLU)
Attention Mask	see AttnGAN
Upsample Block	
Upsampling	Nearest Neighbor
Conv (fs=3, s=1, p=1, BN=1)	X, GLU
Residual Block	
Conv x 2 (fs=3, s=1, p=1, BN=1)	X, GLU, X
Add original input to output of previous conv	
Prepare Label	
Input Shape (Label σ_i)	$(81,)$
Dense (BN=1)	$100, RL$
Reshape	$(100, 1, 1)$
Replicate	$(100, X, X)$
Initial Generator	
Global Pathway Input	noise, sentence emb, layout enc
Input Shape	$(304,)$
Dense (BN=1)	$49152, GLU$
Reshape	$(1536, 4, 4)$
Upsample x 2 (fs=3, s=1, p=1)	768, 384
Object Pathway Input	object labels σ_i
Prepare Label	$(100, 4, 4)$
Upsample x 2 (fs=3, s=1, p=1)	768, 384
Transform with STN	
Concat Pathways	$(768, 16, 16)$
Upsample x 2 (fs=3, s=1, p=1)	192, 96
Output Shape	$(96, 64, 64)$
Generator 128×128	
Global Pathway Input	$(96, 64, 64)$
Input Shape	$(96, 64, 64)$
Attention Mask	$(96, 64, 64)$
Concatenate	$(192, 64, 64)$
Residual x 3	192
Object Pathway Input	object labels σ_i prev G output
Input Shape (Label σ_i)	$(81,), (96, 64, 64)$
Prepare Label	$(128, 16, 16)$
Extr Obj Feat w/ STN	$(96, 16, 16)$
Concatenate	$(224, 16, 16)$
Upsample x 2 (fs=3, s=1, p=1)	192, 96
Transf Obj Feat w/ STN	$(192, 64, 64)$
Concat Pathways	$(288, 64, 64)$
Upsample (fs=3, s=1, p=1)	96
Output Shape	$(96, 128, 128)$
Generator 256×256	
Global Pathway Input	$(96, 128, 128)$
Input Shape	$(96, 128, 128)$
Attention Mask	$(96, 128, 128)$
Concatenate	$(192, 128, 128)$
Residual x 3	192
Object Pathway Input	object labels σ_i prev G output
Input Shape (Label σ_i)	$(81,), (96, 128, 128)$
Prepare Label	$(128, 32, 32)$
Extr Obj Feat w/ STN	$(96, 32, 32)$
Concatenate	$(224, 32, 32)$
Upsample x 2 (fs=3, s=1, p=1)	192, 96
Transf Obj Feat w/ STN	$(192, 128, 128)$
Concat Pathways	$(288, 128, 128)$
Upsample (fs=3, s=1, p=1)	96
Conv (fs=3, s=1, p=1, BN=1)	$3, Tanh$
Output Shape	$(3, 256, 256)$

Table B.7: Overview of the individual layers used in our networks to generate images of resolution 256×256 pixels. Values in brackets (C, H, W) represent the tensor's shape. Numbers in the columns after convolutional, residual, or dense layers describe the number of filters / units in that layer. (fs=x, s=y, p=z, BN=B) describes the filter size, stride, padding, and batch norm for that convolutional / residual layer. Everything not specifically mentioned or explained (e.g. RNN-Encoder, DAMSM) is the same as in the AttnGAN [Xu et al., 2018b].

Learning Rate	0.0002
Training Epochs	120
Batch Size	24
Z-Dim / Img-Caption-Dim	100 / 256
Layout Encoder	
Input Shape	$(100, 16, 16)$
Conv (fs=3, s=2, p=1, BN=0)	$50, LR$
Conv x 2 (fs=3, s=2, p=1, BN=1)	$25, LR$, $12, LR$
Output Shape	$(12, 2, 2)$
Discriminator 64×64	
Global Pathway	
Input Shape	$(3, 64, 64)$
Conv (fs=4, s=2, p=1, BN=0)	$96, LR$
Conv (fs=4, s=2, p=1, BN=1)	$192, LR$
Output Shape	$(192, 16, 16)$
Object Pathway	
Input Shape	$(3, 64, 64)$
Extract Object Feat w/ STN	$(3, 16, 16)$
Concatenate with labels σ_i	$(84, 16, 16)$
Conv (fs=4, s=1, p=1)	$192, LR$
Transform Object Feat w/ STN	$(192, 16, 16)$
Output Shape	$(192, 16, 16)$
Concat Pathways	$(384, 16, 16)$
Conv x 2 (fs=4, s=2, p=1, BN=1)	$384, LR$, $768, LR$
Concat w/ Sentence Embedding	$(1024, 4, 4)$
Conv (fs=3, s=1, p=1, BN=1)	$768, LR$
Conv (fs=4, s=4, p=1, BN=1)	$1, Sigmoid$
Discriminator 128×128	
Global Pathway	
Input Shape	$(3, 128, 128)$
Conv (fs=4, s=2, p=1, BN=0)	$96, LR$
Conv (fs=4, s=2, p=1, BN=1)	$192, LR$
Output Shape	$(192, 32, 32)$
Object Pathway	
Input Shape	$(3, 128, 128)$
Extract Object Feat w/ STN	$(3, 32, 32)$
Concatenate with labels σ_i	$(84, 32, 32)$
Conv (fs=4, s=1, p=1)	$192, LR$
Transform Object Feat w/ STN	$(192, 32, 32)$
Output Shape	$(192, 32, 32)$
Concat Pathways	$(384, 32, 32)$
Conv x 4 (fs=4, s=2, p=1, BN=1)	$384, LR$, $768, LR$ $1536, LR$, $768, LR$
Concat w/ Sentence Embedding	$(1024, 4, 4)$
Conv (fs=3, s=1, p=1, BN=1)	$768, LR$
Conv (fs=4, s=4, p=1, BN=1)	$1, Sigmoid$
Discriminator 256×256	
Global Pathway	
Input Shape	$(3, 256, 256)$
Conv (fs=4, s=2, p=1, BN=0)	$96, LR$
Conv (fs=4, s=2, p=1, BN=1)	$192, LR$
Output Shape	$(192, 64, 64)$
Object Pathway	
Input Shape	$(3, 256, 256)$
Extract Object Feat w/ STN	$(3, 64, 64)$
Concatenate with labels σ_i	$(84, 64, 64)$
Conv (fs=4, s=1, p=1)	$192, LR$
Transform Object Feat w/ STN	$(192, 64, 64)$
Output Shape	$(192, 64, 64)$
Concat Pathways	$(384, 64, 64)$
Conv x 6 (fs=4, s=2, p=1, BN=1)	$384, LR$, $768, LR$ $1536, LR$, $3072, LR$ $1536, LR$ $768, LR$
Concat w/ Sentence Embedding	$(1024, 4, 4)$
Conv (fs=3, s=1, p=1, BN=1)	$768, LR$
Conv (fs=4, s=4, p=1, BN=1)	$1, Sigmoid$

Table B.8: Results of YOLOv3 detections on generated and original images. Recall provides the fraction of images in which YOLOv3 detected the given object. *IoU* (Intersection over Union) measures the maximum IoU per image in which the given object was detected. No ground truth information besides the caption was used for all measurements.

Label	Orig. Img.		AttnGAN	AttnGAN + OP		DM-GAN	Obj-GAN		OP-GAN (Ours)	
	Recall	IoU	Recall	Recall	IoU	Recall	Recall	IoU	Recall	IoU
Person	0.953	0.624	0.698	0.730	0.357	0.840	0.708	0.640	**0.793**	0.289
Dining Table	0.379	0.566	0.104	0.061	0.453	0.094	0.031	0.600	**0.157**	0.495
Cat	0.868	0.644	0.734	0.697	0.264	**0.790**	0.632	0.653	0.656	0.339
Dog	0.813	0.610	0.651	0.778	0.323	0.764	0.846	0.695	**0.850**	0.355
Train	0.826	0.627	0.491	**0.654**	0.370	0.463	0.641	0.670	0.561	0.377
Bus	0.848	0.651	0.615	0.665	0.511	0.766	0.685	0.804	**0.793**	0.366
Clock	0.900	0.502	0.469	0.184	0.359	0.528	**0.649**	0.587	0.587	0.077
Giraffe	0.949	0.662	0.581	0.725	0.486	0.829	0.679	0.585	**0.868**	0.368
Pizza	0.876	0.630	0.793	0.847	0.363	0.883	0.683	0.737	**0.893**	0.449
Horse	0.891	0.611	0.650	0.723	0.528	**0.827**	0.634	0.685	**0.827**	0.328
Elephant	0.937	0.647	0.373	0.653	0.522	**0.705**	0.476	0.737	0.665	0.360
Zebra	0.915	0.650	0.902	0.882	0.420	0.909	0.921	0.735	**0.931**	0.407
Bed	0.732	0.601	0.704	0.661	0.472	**0.796**	0.655	0.573	0.754	0.444
Boat	0.736	0.502	0.211	0.284	0.208	0.244	0.136	0.557	**0.323**	0.198
Toilet	0.912	0.591	0.281	0.325	0.315	0.178	0.382	0.750	**0.543**	0.238
Bird	0.797	0.551	0.358	0.430	0.284	**0.637**	0.546	0.612	0.554	0.267
Skateboard	0.822	0.427	0.040	0.119	0.126	0.153	**0.164**	0.536	0.127	0.116
Car	0.752	0.488	0.143	0.202	0.124	**0.336**	0.196	0.430	0.310	0.102
Bench	0.760	0.547	0.107	0.079	0.311	0.216	**0.339**	0.637	0.259	0.225
Laptop	0.876	0.617	0.071	0.252	0.337	0.229	0.027	0.425	**0.349**	0.323
Surfboard	0.794	0.414	0.140	0.091	0.218	0.225	0.117	0.548	**0.321**	0.172
Truck	0.835	0.631	0.472	0.524	0.442	0.622	0.413	0.685	**0.634**	0.341
Umbrella	0.884	0.548	0.074	0.150	0.177	0.292	0.230	0.591	**0.381**	0.213
Kite	0.822	0.410	0.291	0.163	0.310	0.302	0.370	0.384	**0.414**	0.160
Sports Ball	0.507	0.161	0.112	0.064	0.027	**0.295**	0.198	0.297	0.165	0.004
Cake	0.726	0.570	**0.471**	0.365	0.206	0.385	0.286	0.626	0.423	0.305
Cow	0.886	0.598	0.425	0.566	0.472	**0.649**	0.365	0.638	0.614	0.341
Bicycle	0.686	0.546	0.281	0.251	0.284	**0.498**	0.249	0.503	**0.486**	0.297
Chair	0.717	0.566	0.175	0.142	0.157	**0.269**	0.070	0.257	0.258	0.130
Frisbee	0.803	0.350	0.025	0.018	0.050	0.061	0.099	0.625	**0.101**	0.025
Bear	0.638	0.637	**0.812**	0.794	0.431	0.800	0.712	0.761	0.737	0.341
Sandwich	0.674	0.630	0.505	0.634	0.310	0.508	0.585	0.568	**0.667**	0.402
Sheep	0.910	0.593	0.303	0.403	0.239	0.545	**0.573**	0.642	0.559	0.251
Vase	0.858	0.600	0.114	0.152	0.468	0.175	0.150	0.306	**0.276**	0.271
Bowl	0.675	0.633	0.315	0.113	0.170	0.212	0.066	0.598	**0.330**	0.216
Sink	0.712	0.431	0.075	0.128	0.127	0.144	0.165	0.340	**0.184**	0.102
Stop Sign	0.874	0.608	0.183	0.225	0.207	0.522	0.510	0.830	**0.591**	0.292
Banana	0.788	0.578	0.552	**0.593**	0.208	0.433	0.287	0.572	0.444	0.308
Monitor	0.754	0.594	0.278	0.225	0.477	0.445	0.385	0.759	**0.606**	0.213
Skis	0.576	0.315	0.010	0.023	0.057	0.023	0.023	0.512	**0.040**	0.146
Hot Dog	0.711	0.621	0.404	0.355	0.227	0.452	0.371	0.671	**0.592**	0.332
Fire Hydrant	0.927	0.613	0.414	0.256	0.388	0.420	0.274	0.666	**0.426**	0.282
Sofa	0.834	0.584	0.253	0.179	0.259	**0.397**	0.221	0.470	0.331	0.292
Teddy Bear	0.806	0.643	0.637	**0.688**	0.336	0.615	0.410	0.707	0.455	0.328
Aeroplane	0.916	0.575	0.612	0.382	0.211	0.571	0.297	0.513	**0.665**	0.318
Tie	0.800	0.574	0.138	0.074	**0.157**	0.095	0.113	0.385	0.117	0.117
Tennis Racket	0.830	0.432	0.019	0.044	0.071	0.048	**0.141**	0.518	0.058	0.093

Cell Phone	0.590	0.513	0.036	0.054	0.067	0.105	0.134	0.563	**0.264**	0.201
Refrigerator	0.881	0.631	**0.593**	0.252	0.408	0.456	0.409	0.518	0.558	0.375
Cup	0.706	0.586	0.061	0.054	0.022	**0.131**	0.040	0.430	0.067	0.137
Broccoli	0.756	0.575	0.130	0.137	0.240	0.255	0.248	0.601	**0.528**	0.267
Donut	0.854	0.655	0.076	0.089	0.213	0.138	0.207	0.712	**0.304**	0.297
Bottle	0.782	0.590	0.072	0.047	0.020	**0.148**	0.027	0.002	0.069	0.053
Suitcase	0.851	0.612	0.049	0.043	0.407	0.070	0.118	0.662	**0.122**	0.318
Snowboard	0.746	0.411	0.055	0.030	0.085	**0.101**	0.080	0.356	0.073	0.114
Book	0.628	0.500	0.006	0.032	0.340	**0.064**	0.007	0.390	0.051	0.184
Remote	0.619	0.440	0.014	0.015	0.120	0.123	0.044	0.488	0.038	0.133
Traffic Light	0.942	0.450	0.607	0.565	0.409	**0.724**	0.619	0.559	0.653	0.215
Keyboard	0.783	0.495	0.397	0.083	0.064	**0.687**	0.095	0.701	0.350	0.154
Apple	0.588	0.593	0.054	0.021	0.162	0.119	0.121	0.709	**0.140**	0.237
Oven	0.699	0.606	0.067	0.074	0.520	0.174	0.055	0.338	**0.213**	0.304
Motorcycle	0.910	0.597	0.422	0.409	0.396	0.476	0.206	0.528	**0.629**	0.363
Carrot	0.590	0.537	0.081	0.045	0.097	0.083	0.044	0.545	**0.116**	0.153
Scissor	0.654	0.616	0.047	0.066	0.238	0.071	0.024	0.242	**0.116**	0.261
Parking Meter	0.816	0.600	0.222	0.114	0.323	0.535	**0.658**	0.759	0.582	0.300
Microwave	0.849	0.568	0.066	0.027	0.326	**0.120**	0.062	0.518	0.074	0.144
Orange	0.826	0.617	0.024	0.113	0.104	0.303	0.084	0.677	**0.406**	0.208
Knife	0.577	0.537	0.017	0.018	0.085	0.015	0.011	0.148	**0.037**	0.069
Fork	0.675	0.574	0.029	0.083	0.092	0.029	0.052	0.489	**0.095**	0.124
Baseball Bat	0.653	0.397	0.018	0.010	0.022	0.011	**0.105**	0.395	0.021	0.078
Toothbrush	0.557	0.505	0.036	0.102	0.157	0.025	**0.107**	0.554	0.091	0.160
Wine Glass	0.888	0.583	**0.194**	0.119	0.086	0.177	0.076	0.275	0.111	0.110
Backpack	0.620	0.529	0.024	0.049	0.000	**0.107**	0.093	0.000	0.041	0.000
Spoon	0.545	0.533	0.069	0.066	0.000	0.055	0.032	0.000	**0.091**	0.000
Handbag	0.537	0.583	0.014	0.014	0.000	0.042	0.021	0.000	**0.043**	0.000
Toaster	0.093	0.598	0.000	0.000	0.000	**0.004**	0.000	0.000	0.000	0.000
Potted Plant	0.753	0.574	0.068	0.048	0.000	0.092	0.035	0.000	**0.112**	0.000
Mouse	0.804	0.537	0.076	0.024	0.067	**0.145**	0.095	0.636	0.096	0.167
Baseball Glove	0.667	0.514	0.006	0.006	0.001	0.042	**0.083**	0.591	0.020	0.198
Hair Drier	0.050	0.158	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table B.9: Results of YOLOv3 detections on ablations of our model. Recall provides the fraction of images in which YOLOv3 detected the given object. *IoU* (Intersection over Union) measures the maximum IoU per image in which the given object was detected. No ground truth information besides the caption was used for all measurements.

Label	*OPv2*		*OPv2 + BBL*		*OPv2 + MO*		*OPv2 + BBL + MO*	
	Recall	IoU	Recall	IoU	Recall	IoU	Recall	IoU
Person	0.783	0.279	0.769	0.278	0.789	0.288	0.771	0.286
Dining Table	0.095	0.462	0.126	0.453	0.106	0.466	0.106	0.467
Cat	0.699	0.336	0.725	0.330	0.702	0.337	0.697	0.330
Dog	0.831	0.342	0.790	0.330	0.745	0.330	0.827	0.351
Train	0.645	0.390	0.699	0.388	0.642	0.389	0.654	0.379
Bus	0.756	0.372	0.721	0.372	0.785	0.384	0.802	0.361
Clock	0.489	0.096	0.542	0.130	0.542	0.098	0.401	0.097

Giraffe	0.796	0.337	0.853	0.356	0.819	0.353	0.831	0.365
Pizza	0.853	0.428	0.883	0.427	0.837	0.433	0.822	0.437
Horse	0.769	0.313	0.774	0.315	0.789	0.331	0.789	0.327
Elephant	0.684	0.368	0.722	0.373	0.658	0.356	0.646	0.363
Zebra	0.946	0.393	0.953	0.404	0.955	0.396	0.941	0.406
Bed	0.806	0.457	0.742	0.466	0.747	0.456	0.765	0.464
Boat	0.315	0.207	0.224	0.196	0.244	0.232	0.290	0.214
Toilet	0.523	0.252	0.533	0.246	0.455	0.256	0.473	0.250
Bird	0.610	0.258	0.650	0.261	0.628	0.249	0.619	0.264
Skateboard	0.162	0.081	0.156	0.076	0.097	0.095	0.131	0.113
Car	0.274	0.119	0.236	0.109	0.198	0.129	0.286	0.112
Bench	0.240	0.229	0.180	0.228	0.255	0.236	0.256	0.225
Laptop	0.324	0.309	0.237	0.293	0.201	0.299	0.298	0.317
Surfboard	0.268	0.149	0.215	0.152	0.266	0.144	0.266	0.170
Truck	0.585	0.341	0.560	0.333	0.590	0.343	0.593	0.338
Umbrella	0.130	0.178	0.189	0.183	0.163	0.187	0.219	0.210
Kite	0.354	0.120	0.518	0.123	0.340	0.104	0.427	0.157
Cake	0.448	0.280	0.424	0.295	0.510	0.305	0.486	0.309
Sports Ball	0.067	0.005	0.095	0.004	0.192	0.004	0.128	0.004
Cow	0.611	0.298	0.623	0.324	0.621	0.332	0.645	0.340
Bicycle	0.401	0.280	0.368	0.245	0.447	0.283	0.472	0.290
Chair	0.138	0.133	0.150	0.134	0.250	0.141	0.262	0.138
Frisbee	0.066	0.024	0.052	0.029	0.043	0.035	0.063	0.022
Bear	0.749	0.348	0.754	0.351	0.739	0.345	0.758	0.354
Sandwich	0.648	0.380	0.656	0.380	0.613	0.390	0.716	0.394
Sheep	0.617	0.245	0.578	0.217	0.573	0.247	0.596	0.254
Vase	0.182	0.181	0.187	0.210	0.154	0.204	0.239	0.220
Bowl	0.238	0.223	0.215	0.202	0.298	0.223	0.300	0.213
Sink	0.196	0.131	0.172	0.106	0.206	0.090	0.195	0.113
Stop Sign	0.584	0.279	0.453	0.280	0.494	0.237	0.449	0.270
Banana	0.464	0.274	0.517	0.289	0.426	0.280	0.504	0.284
Monitor	0.510	0.209	0.502	0.225	0.535	0.222	0.581	0.225
Hotdog	0.481	0.297	0.443	0.305	0.478	0.311	0.576	0.322
Skis	0.021	0.111	0.037	0.121	0.042	0.115	0.039	0.127
Sofa	0.274	0.304	0.289	0.284	0.324	0.334	0.270	0.309
Fire Hydrant	0.456	0.290	0.411	0.298	0.386	0.311	0.360	0.295
Teddy Bear	0.595	0.339	0.608	0.344	0.582	0.350	0.621	0.341
Aeroplane	0.701	0.315	0.642	0.321	0.599	0.283	0.634	0.310
Tie	0.132	0.098	0.085	0.120	0.112	0.103	0.088	0.103
Tennis Racket	0.044	0.073	0.042	0.066	0.070	0.087	0.041	0.095

Cell Phone	0.199	0.156	0.100	0.137	0.147	0.150	0.189	0.171
Refrigerator	0.435	0.373	0.438	0.376	0.522	0.366	0.539	0.366
Cup	0.047	0.083	0.078	0.111	0.060	0.091	0.078	0.134
Broccoli	0.418	0.243	0.468	0.243	0.448	0.232	0.436	0.261
Donut	0.234	0.247	0.248	0.246	0.277	0.287	0.305	0.278
Bottle	0.079	0.023	0.076	0.009	0.103	0.028	0.140	0.026
Suitcase	0.100	0.271	0.084	0.289	0.129	0.308	0.117	0.296
Book	0.015	0.141	0.022	0.135	0.033	0.148	0.041	0.142
Snowboard	0.085	0.102	0.067	0.105	0.072	0.106	0.089	0.127
Remote	0.060	0.080	0.066	0.096	0.089	0.120	0.051	0.141
Traffic Light	0.696	0.160	0.608	0.163	0.703	0.164	0.600	0.175
Keyboard	0.375	0.147	0.494	0.147	0.484	0.167	0.375	0.156
Oven	0.141	0.304	0.172	0.308	0.213	0.322	0.206	0.317
Apple	0.227	0.215	0.160	0.163	0.164	0.199	0.160	0.217
Motorcycle	0.590	0.376	0.515	0.335	0.420	0.346	0.501	0.345
Scissors	0.045	0.196	0.105	0.264	0.079	0.271	0.119	0.239
Carrot	0.080	0.151	0.093	0.163	0.106	0.160	0.106	0.155
Parking Meter	0.553	0.300	0.305	0.263	0.449	0.288	0.481	0.327
Microwave	0.150	0.184	0.120	0.190	0.080	0.184	0.078	0.142
Orange	0.323	0.186	0.335	0.179	0.308	0.201	0.314	0.195
Knife	0.035	0.090	0.040	0.099	0.028	0.095	0.036	0.053
Fork	0.096	0.084	0.098	0.118	0.067	0.088	0.095	0.121
Baseball Bat	0.016	0.056	0.022	0.038	0.019	0.047	0.039	0.050
Toothbrush	0.041	0.170	0.082	0.204	0.063	0.172	0.094	0.181
Wine Glass	0.153	0.101	0.160	0.090	0.155	0.080	0.145	0.110
Backpack	0.049	0.000	0.020	0.000	0.045	0.000	0.051	0.000
Spoon	0.093	0.000	0.107	0.000	0.075	0.000	0.078	0.000
Handbag	0.001	0.000	0.015	0.007	0.014	0.000	0.026	0.000
Toaster	0.006	0.000	0.001	0.000	0.000	0.000	0.001	0.000
Mouse	0.073	0.083	0.077	0.103	0.047	0.108	0.084	0.124
Potted Plant	0.065	0.000	0.055	0.000	0.085	0.000	0.075	0.000
Baseball Glove	0.041	0.042	0.029	0.210	0.033	0.211	0.022	0.137
Hair Drier	0.002	0.000	0.000	0.000	0.000	0.000	0.002	0.000

B.4 Learning Representations from a Single Data Point

This section gives more details about our experiments outlined in Section 5.2. Figure B.8 shows more examples of unconditional image generation for both

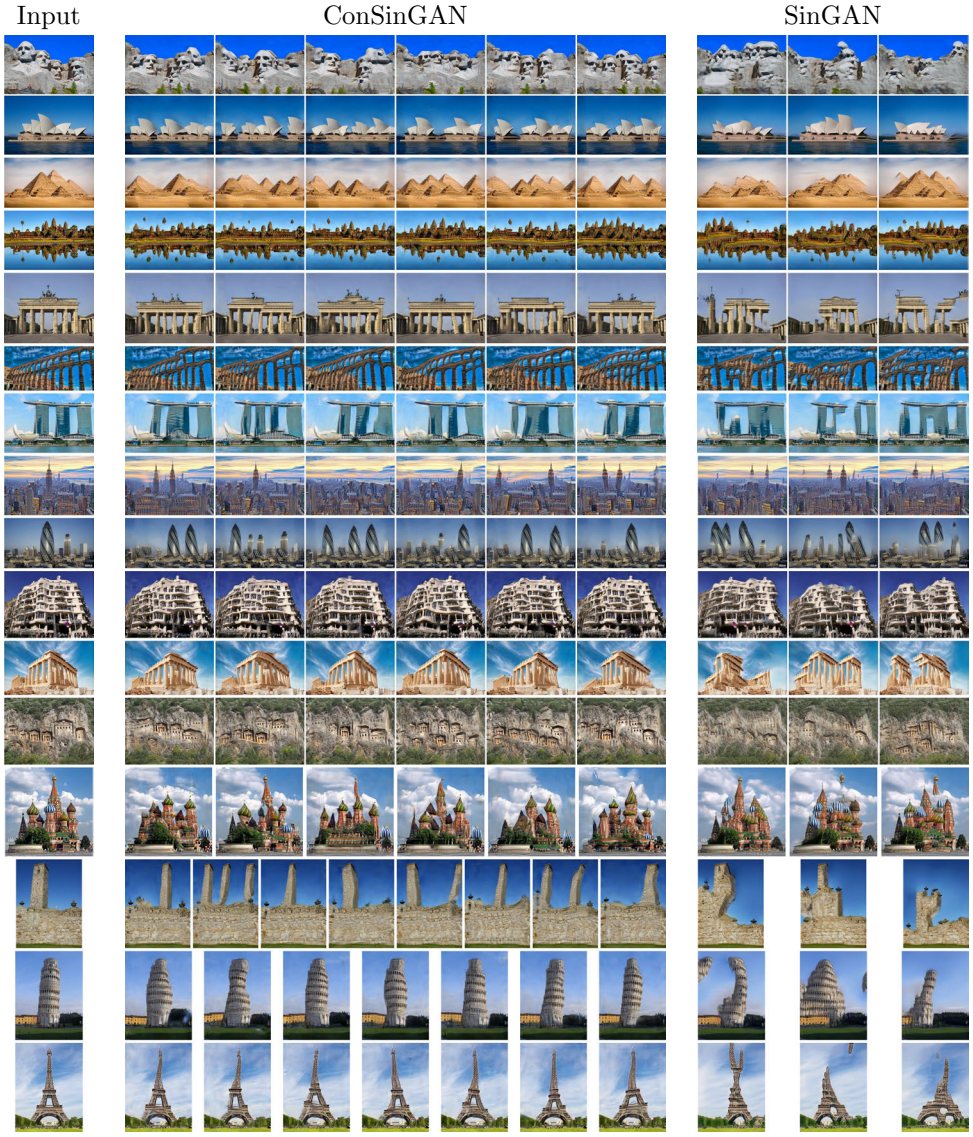


Figure B.8: Unconditional image generation with ConSinGAN and SinGAN. All ConSinGAN models were trained on six stages, all SinGAN models were trained on eight – ten stages. We can see how ConSinGAN is able to model the global image structure better in most cases, despite being trained on fewer stages than SinGAN.

ConSinGAN and SinGAN [Shaham et al., 2019]. Our ConSinGAN models were trained on six stages while the SinGAN models were trained with the default scaling factor of 0.75 for eight – ten stages per image. Despite this we can see that the ConSinGAN is capable of modeling the global image layout better than SinGAN in most cases. Training a ConSinGAN model takes roughly 20-25 minutes on our hardware (NVIDIA GeForce GTX 1080Ti), while training a SinGAN model takes roughly 2 hours.

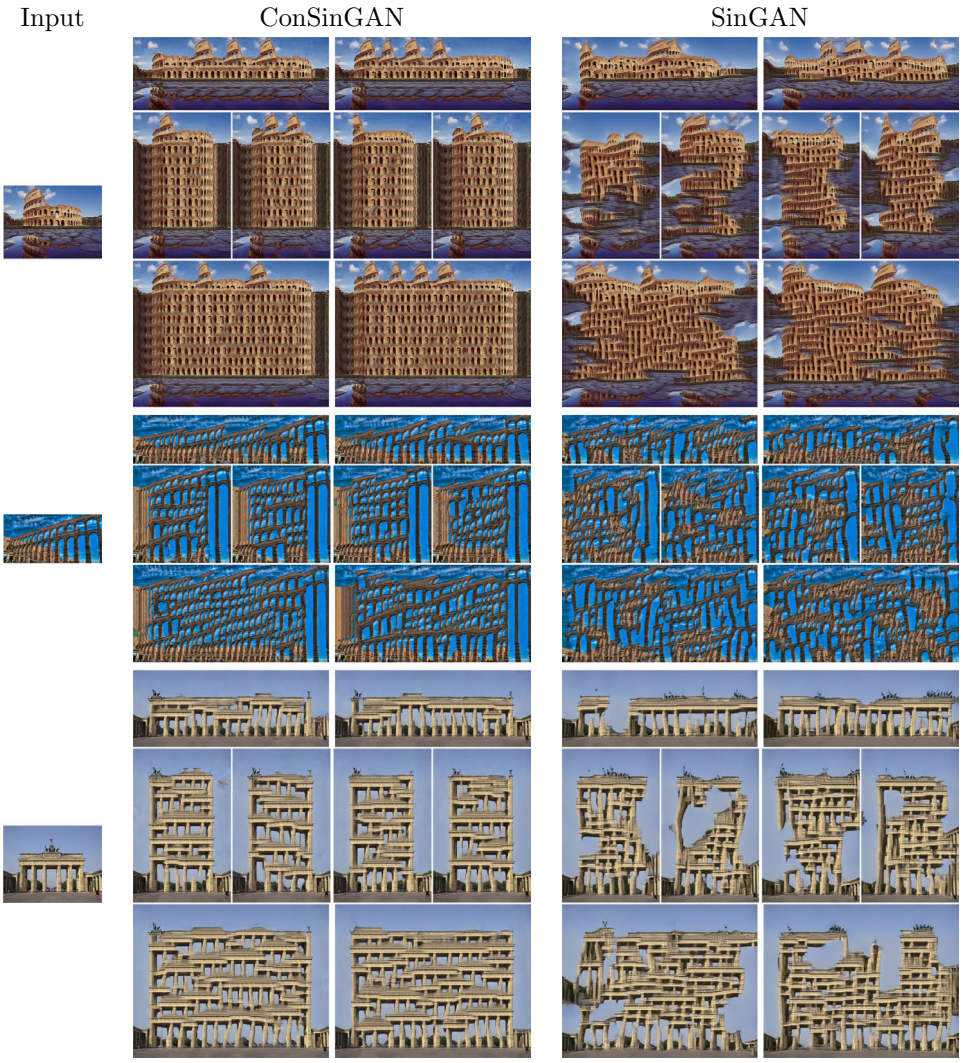


Figure B.9: Unconditional image generation with ConSinGAN and SinGAN. We show images were we scale the input noise map by a factor of two along each side and along both sides. The ConSinGAN models were trained on six stages, while the SinGAN models were trained on eight – ten stages.

Figure B.9 and Figure B.10 show results of unconditional image generation with different aspect ratios and resolutions. Scaling in the horizontal direction usually works better for both models. Scaling in the vertical direction is much more challenging for both models, however, the ConSinGAN handles this better than the SinGAN.

Figure B.10: Unconditional image generation with ConSinGAN and SinGAN. We show images were we scale the input noise map by a factor of two along each side and along both sides. The ConSinGAN models were trained on six stages, while the SinGAN models were trained on eight – ten stages.

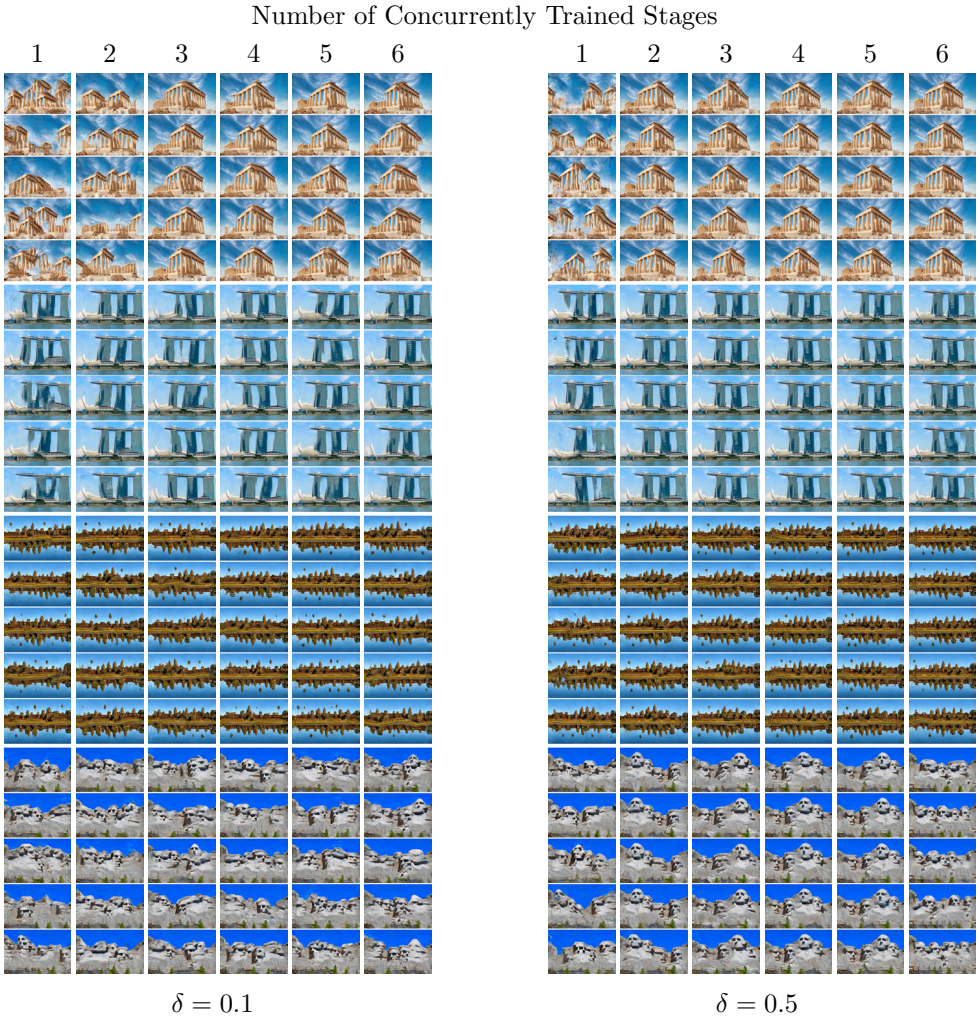


Figure B.11: Interplay between learning rate scaling δ and the number of concurrently trained stages for models that were trained on a total of six stages on different images. We can see how the image diversity usually decreases with increasing δ or increasing number of concurrently trained stages. All images are randomly sampled.

B.4.1 Number of Concurrently Trained Stages and Learning Rate Scaling

Figure B.11 shows more visualizations of the interplay between the learning rate scaling δ and the number of concurrently trained stages. Again, we observe that image diversity decreases with increasing δ and increasing number of concurrently trained stages, leading to complete overfitting when trained end-to-end.

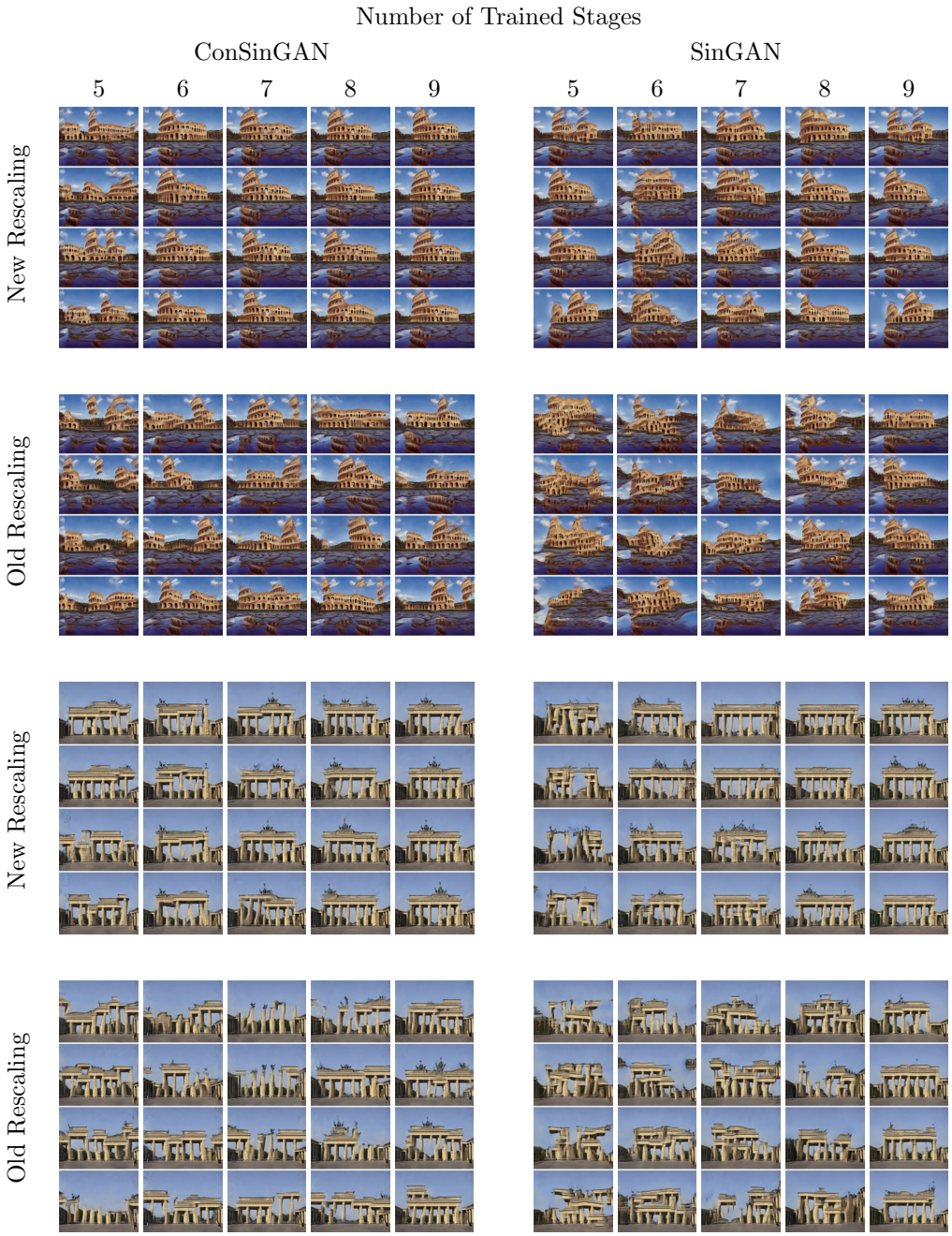


Figure B.12: Comparison between our updated and the original rescaling method. We can see that both models benefit from the updated rescaling method and can generate realistic images with fewer stages. All images are randomly sampled.

B.4.2 Comparison of Original and Improved Rescaling Method

Figure B.12 and Figure B.13 show more examples of how the image quality develops with an increasing number of stages for the original and our improved rescaling

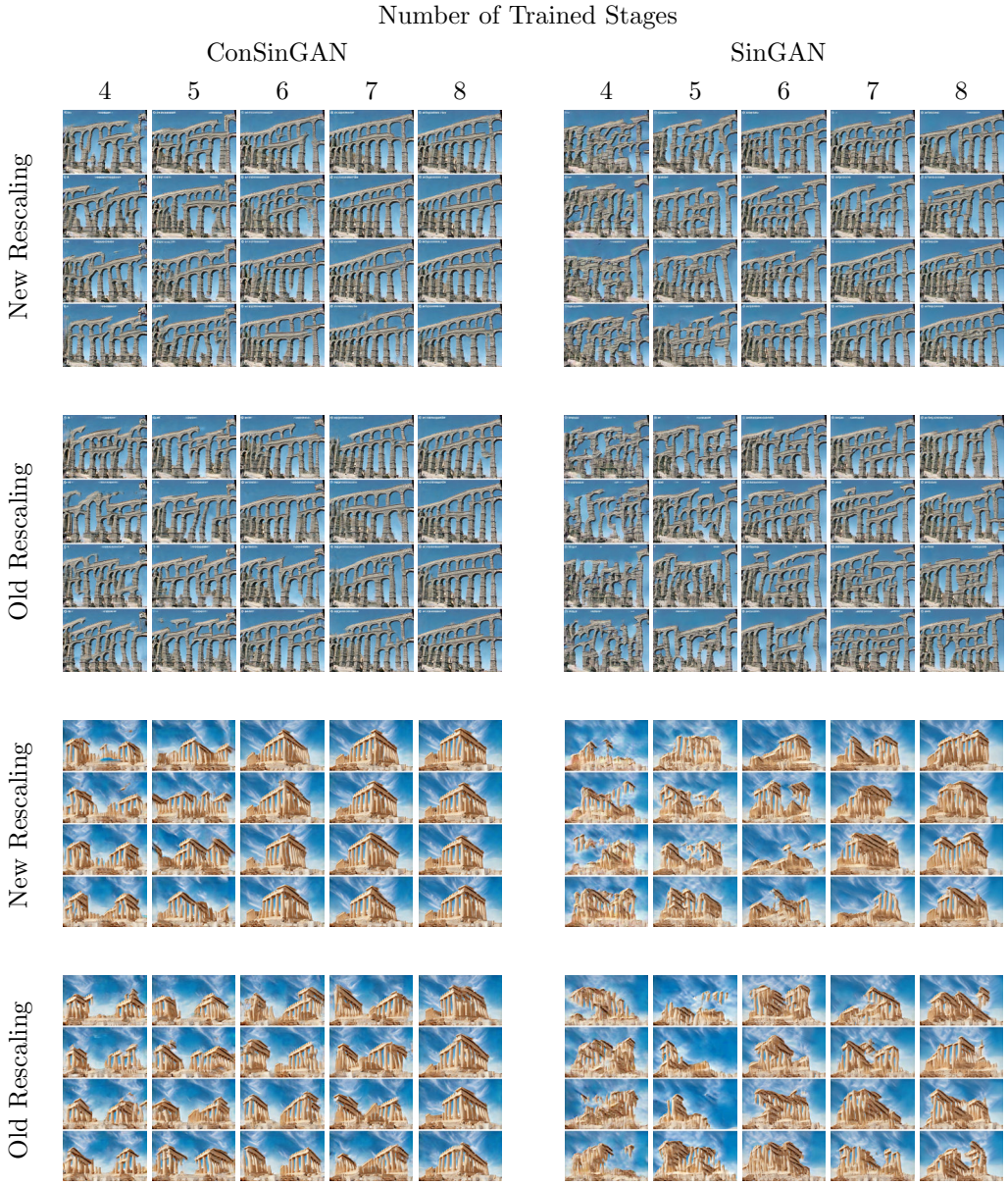


Figure B.13: Comparison between our updated and the original rescaling method. We can see that both models benefit from the updated rescaling method and can generate realistic images with fewer stages. All images are randomly sampled.

method. Note that the ConSinGAN starts to generate realistic images after already 5-6 stages while the SinGAN usually needs more than 7 stages.

B.4.3 Image Harmonization

Figure B.14 shows further comparisons between SinGAN and ConSinGAN on the image harmonization task. We can see that ConSinGAN often performs better despite being trained on fewer stages than SinGAN. SinGAN is also not able to transform color objects into black-and-white images, while this is no problem for ConSinGAN. Figure B.15 and Figure B.16 show more comparisons between ConSinGAN and Deep Painterly Harmonization (DPH) [Luan et al., 2018] on image harmonization tasks of images with higher resolution. Note that DPH needs the target image and mask as input for its algorithm, while ConSinGAN is trained with randomly augmented images and only sees the target image at test time (except for the fine-tuned case).

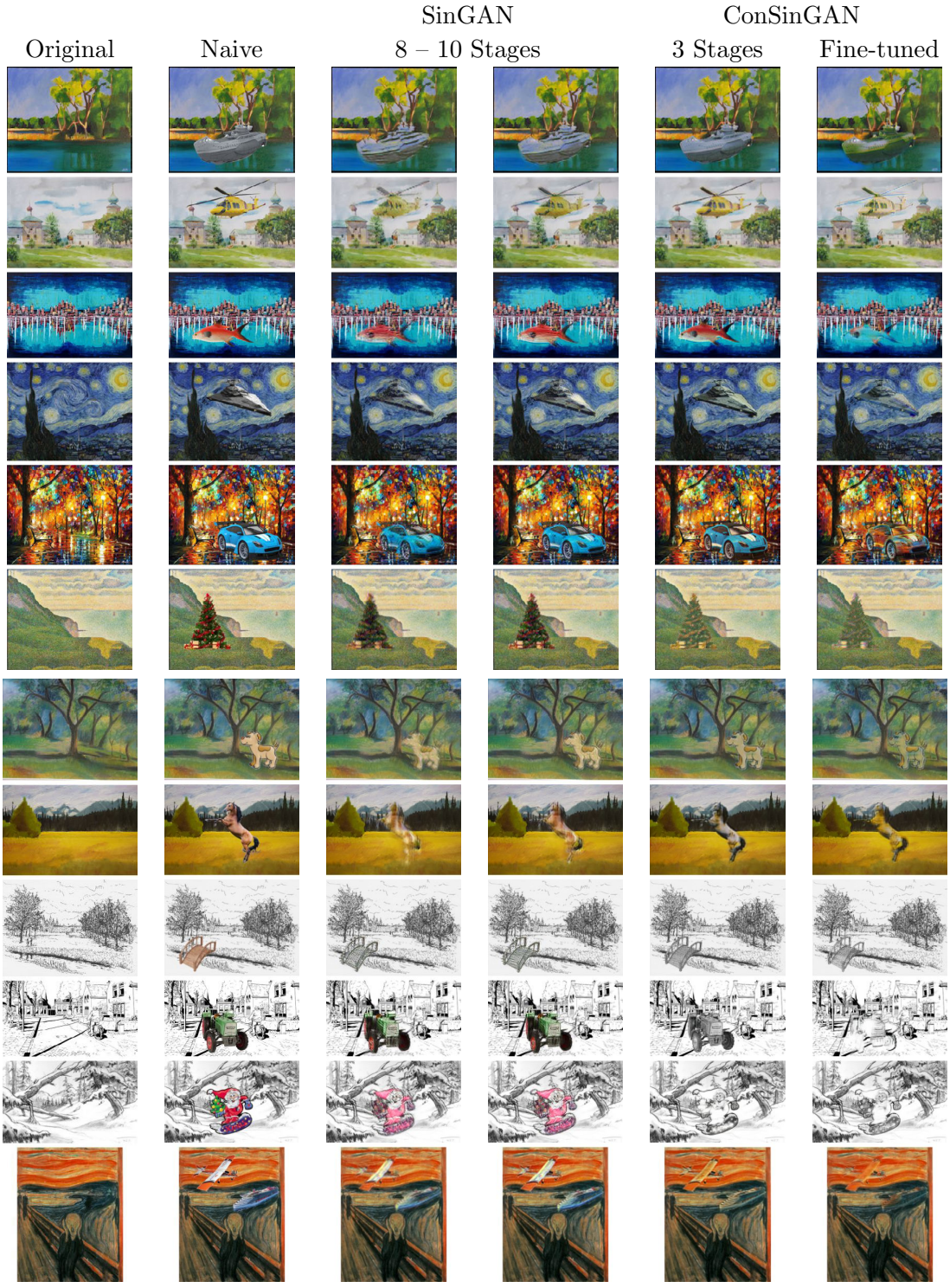


Figure B.14: Comparison of ConSinGAN and SinGAN on image harmonization. Our model often produces better results despite being trained on fewer stages.

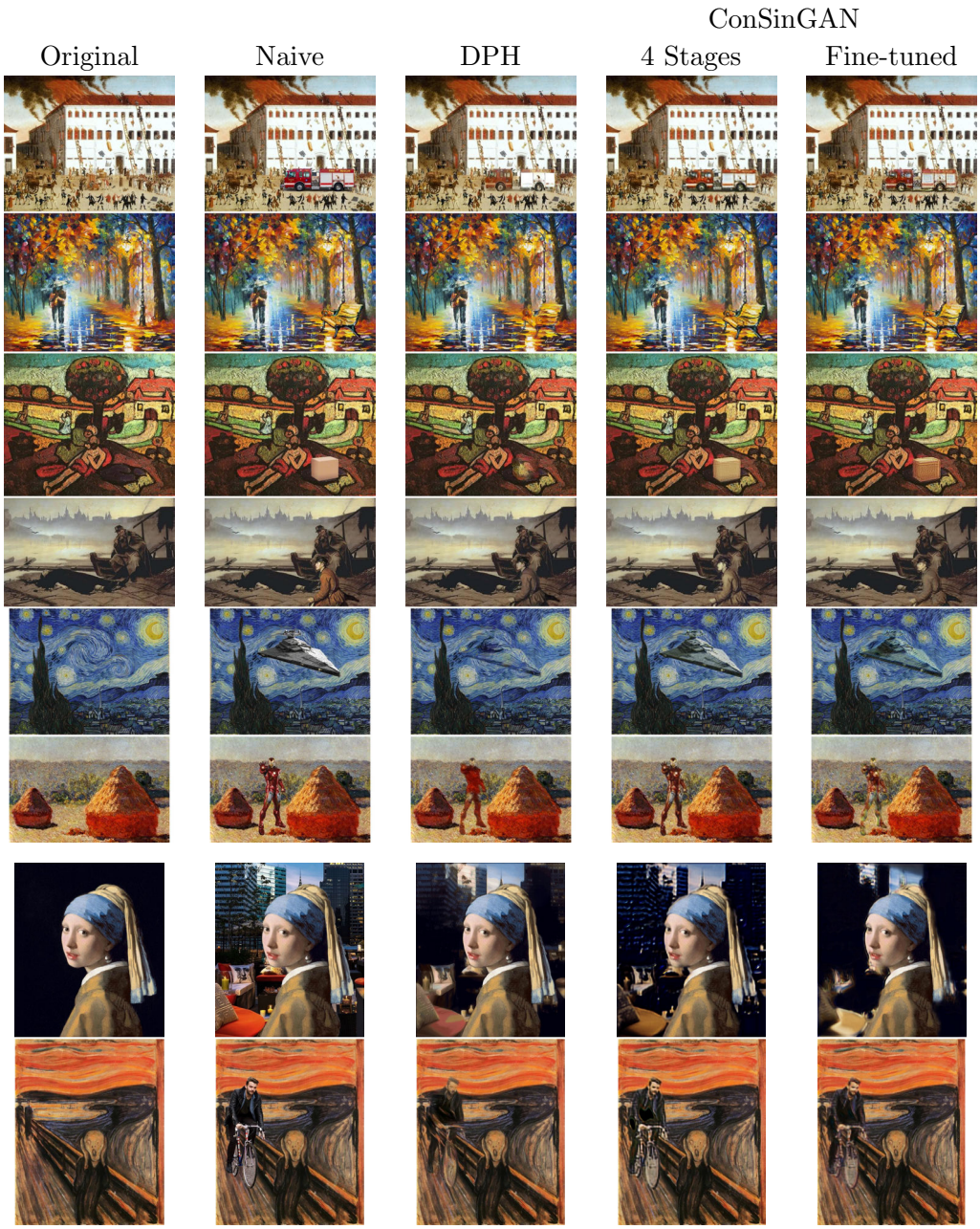


Figure B.15: Comparison of ConSinGAN and Deep Painterly Harmonization (DPH) on high-resolution image harmonization. Images from DPH taken from their official Github implementation. In contrast to DPH, our model does only see the naive image during training if we fine-tune it (last column), but not for our general training procedure (fourth column).

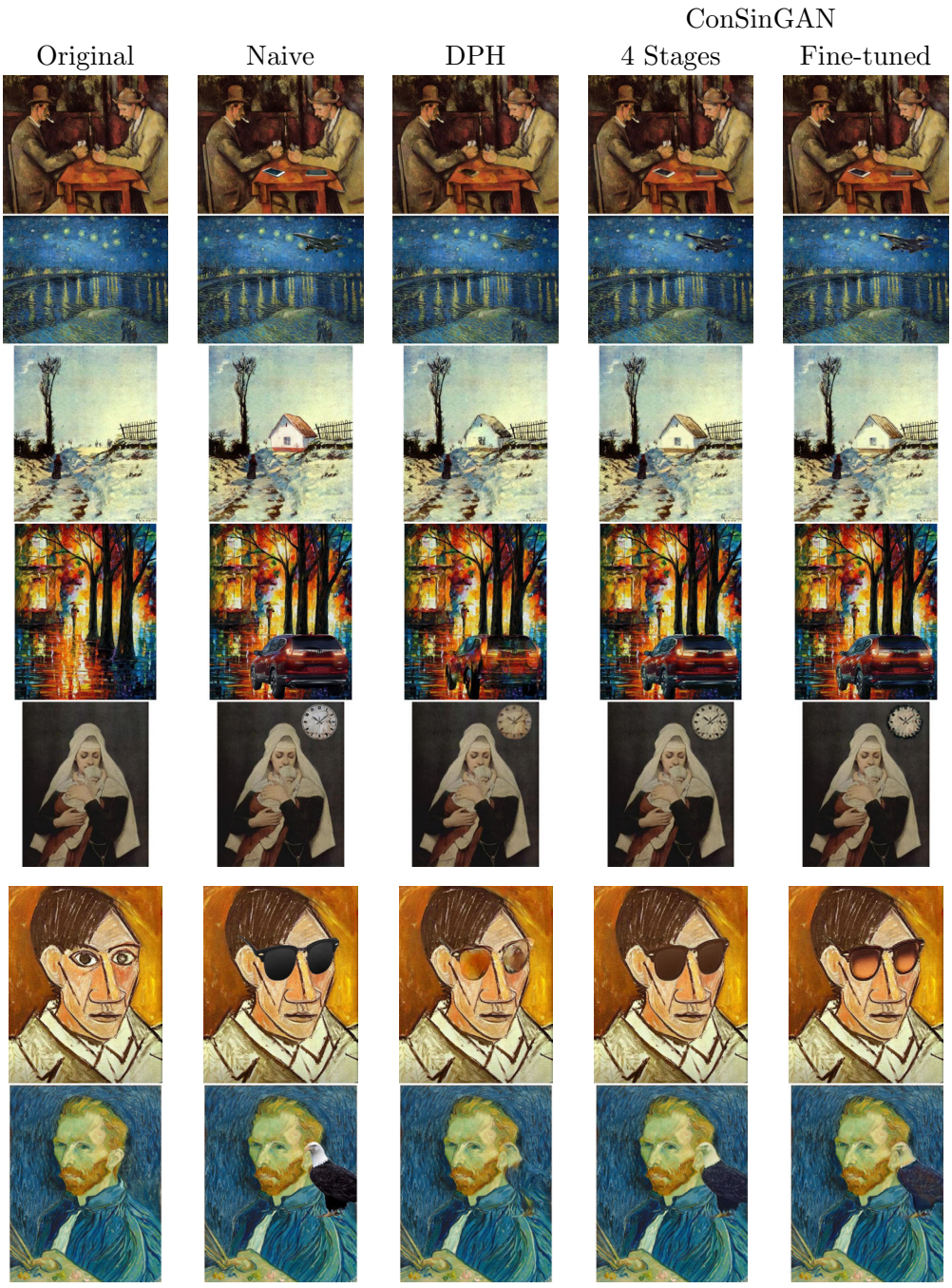


Figure B.16: Comparison of ConSinGAN and Deep Painterly Harmonization (DPH) on high-resolution image harmonization. Images from DPH taken from their official Github implementation. In contrast to DPH, our model does only see the naive image during training if we fine-tune it (last column), but not for our general training procedure (fourth column).

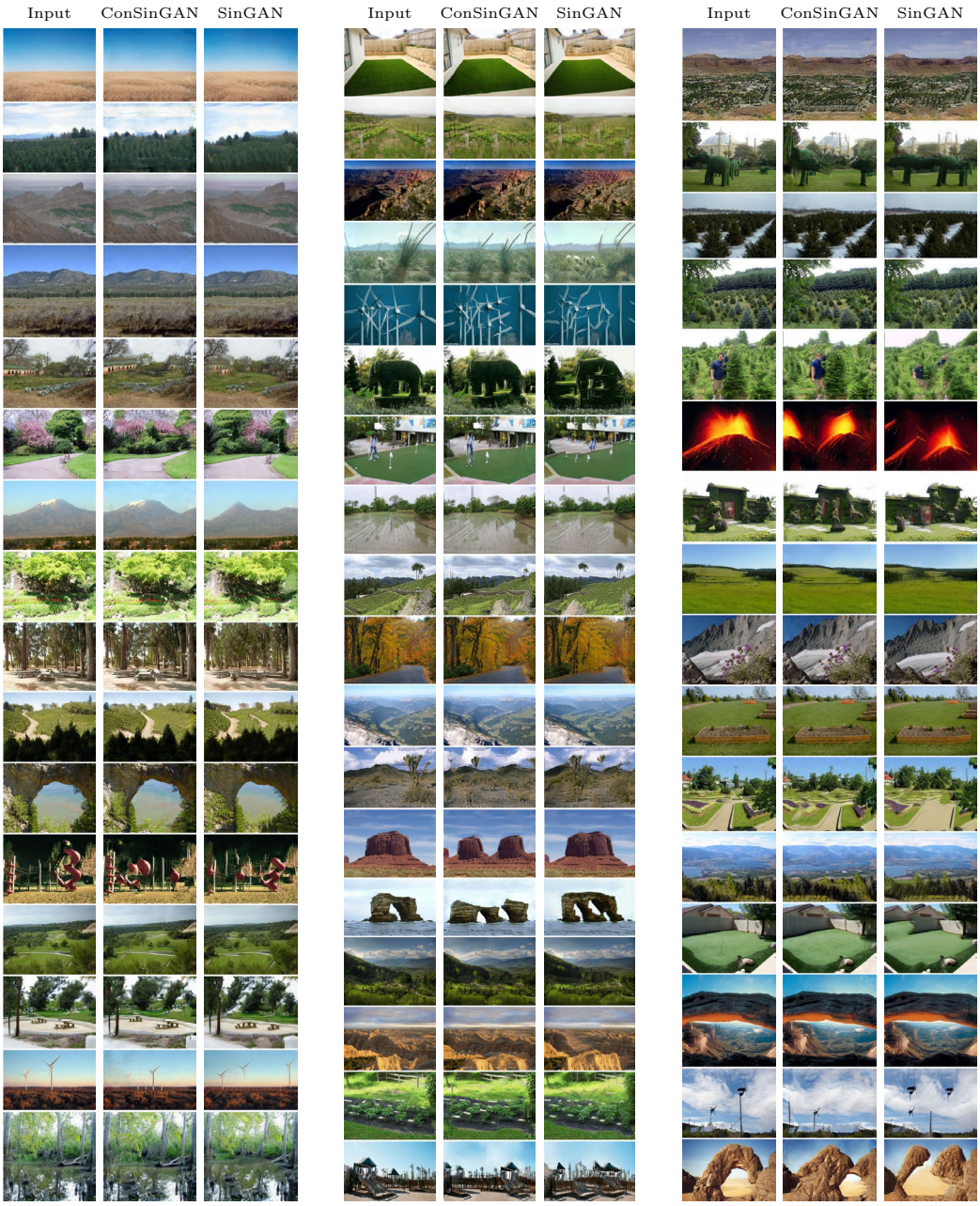


Figure B.17: Images from the 'Places' data set used in our user study.

B.4.4 Images From User Studies

Figure B.17 and Figure B.18 show the images that were used in the two user studies on the 'Places' and 'LSUN' data sets respectively.

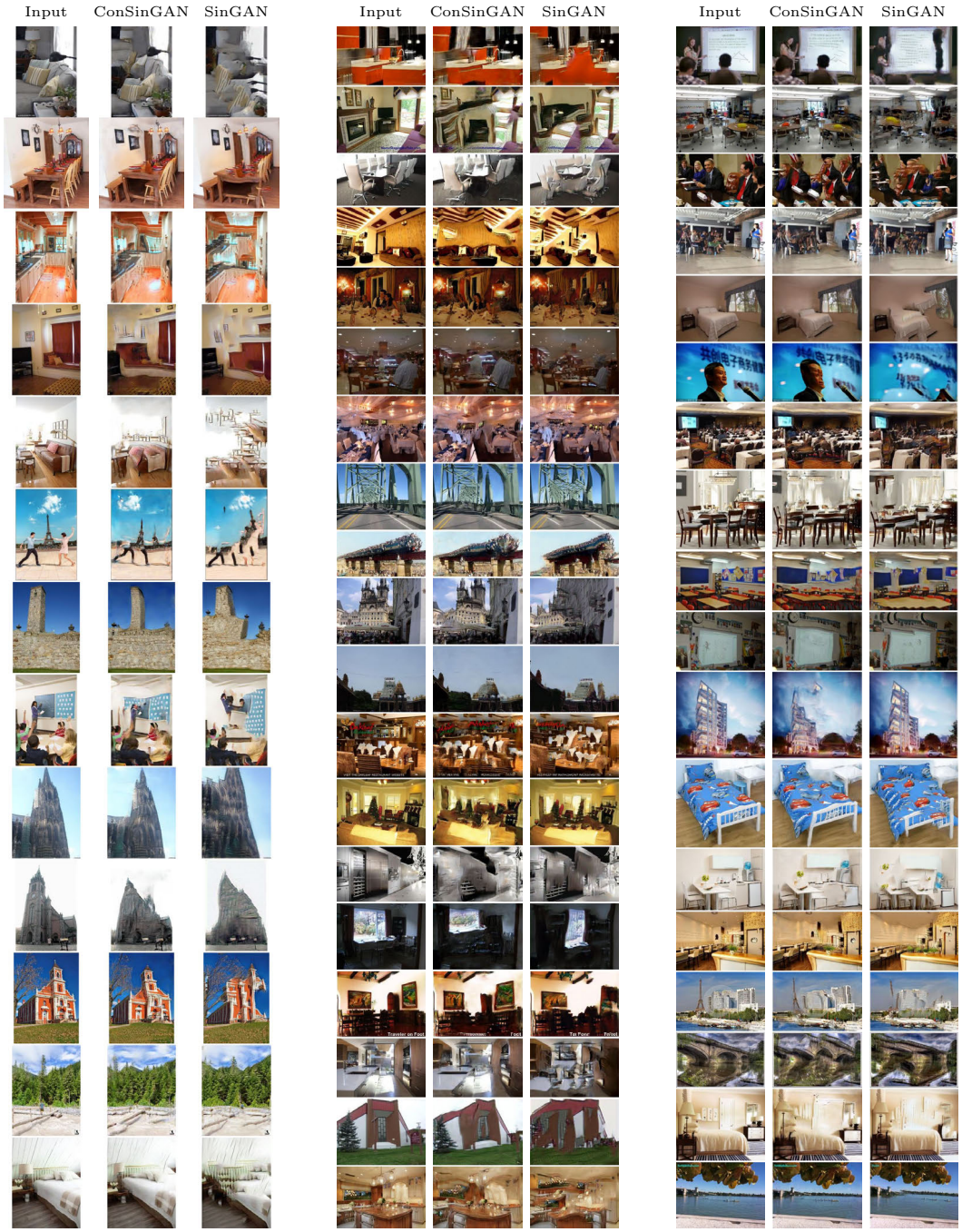


Figure B.18: Images from the 'LSUN' data set used in our user study.

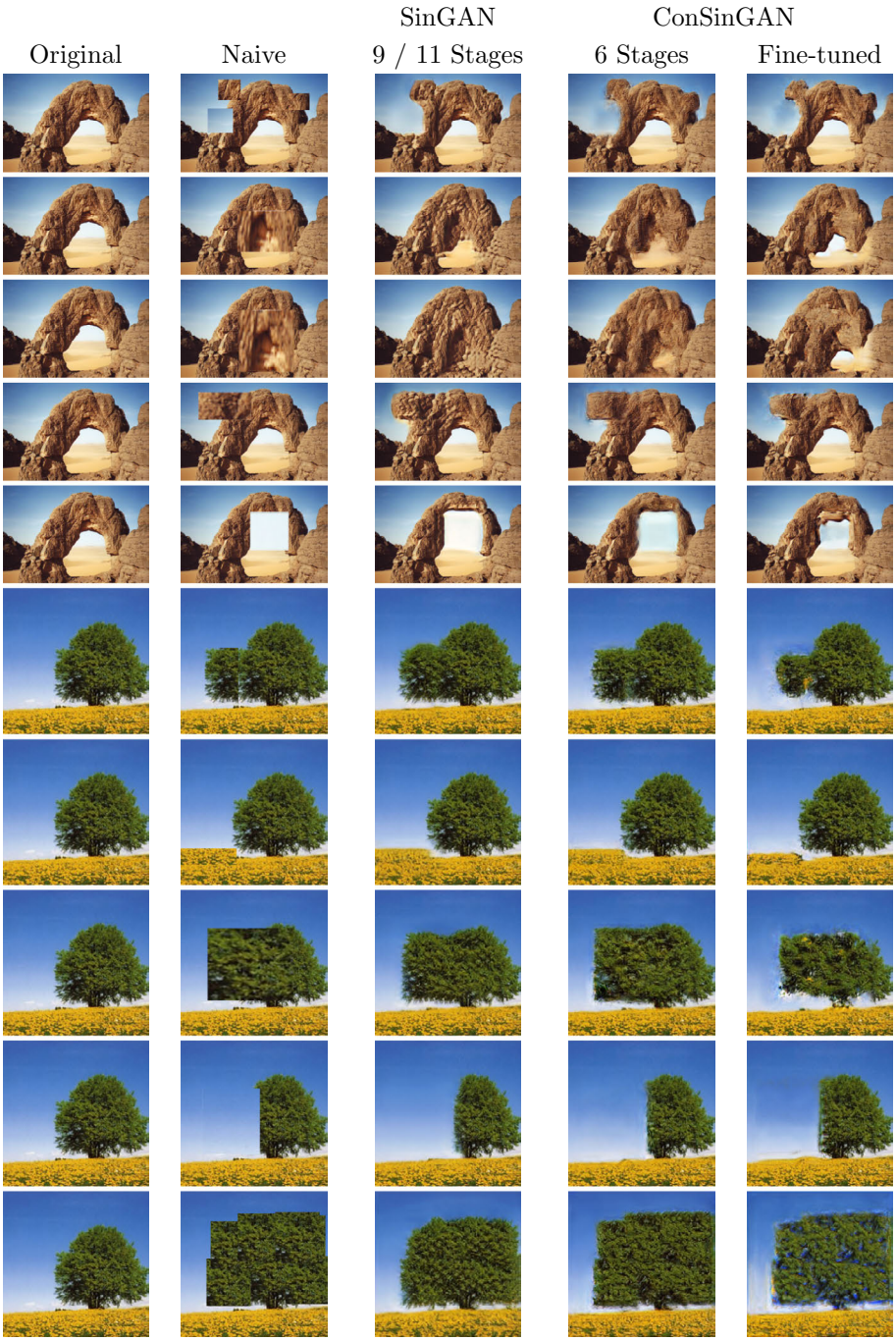


Figure B.19: Editing images with SinGAN and ConSinGAN. Note that the SinGAN model is trained on 9 – 11 stages, while ConSinGAN is only trained for 6 stages

B.4.5 Image Editing

Figure B.19 shows some examples of the image editing task with SinGAN and ConSinGAN. We trained the full SinGAN model (9 and 11 stages respectively) and chose the best results. The ConSinGAN was trained on only six stages and

with only 1,000 iterations per stage, as opposed to 2,000 iterations per stage at the SinGAN. We can observe that both models have strengths and weaknesses. SinGAN is usually better at merging background objects (e.g. the sky in the stone image) but also introduces many artifacts, e.g. it changes the texture of the stone in many cases even in places where no editing took place. Furthermore, its texture is very repetitive when large ares are changed, e.g. the leaves in the changed areas of the tree.

ConSinGAN, on the other hand, does not change the structure in areas that are not edited and exhibits none of the repetitiveness in the features. However, it sometimes fails to merge the background as successfully as SinGAN does, see e.g. again the sky in the stone image. We can also see that ConSinGAN tends to adhere more closely to the layout of the edited image and mainly rounds and smooths the edges along edited areas. SinGAN changes more of the image which can sometimes look more realistic, but might not always be desired if the changes were done carefully and the artist wants the model to adhere to his changes as exactly as possible. Also note that we did not experiment with any hyperparameters for ConSinGAN for the image editing task and it might be possible to achieve better results by finding better hyperparameters.

B.4.6 Other Approaches We Explored

Multiple Discriminators at Each Stage Similar to other work, e.g. InGAN [Shocher et al., 2019], we tried using several discriminators at each stage. This can potentially be helpful for tasks such as unconditional image generation at different resolutions (see Figure B.9 and Figure B.10). To test this we adapted our model so that the generator is trained with multiple discriminators at each stage. We generate images at different fixed resolutions, e.g. by scaling the width and height of the input noise map by factors 0.5, 1.0, 1.5, and 2.0. As a result, the generator generates several images at each iteration which are used as inputs for several discriminators (one for each resolution).

Each of the discriminators is trained on only one specific resolution where the 'real' image is a rescaled version of the original image. We observed that this does indeed often lead to improved results for the unconditional generation of various resolutions. However, for other applications, the results were inconsistent – sometimes it improved things, sometimes it did not. Since training several discriminators at each stage increases the training time considerably we decided to not use this approach in the default settings of our method.

Adaptive Number of Training Iterations at Each Stage At the moment, each stage is trained for a pre-defined number of iterations, e.g. 2,000 iterations for unconditional image generation and 1,000 iterations for image harmonization. We believe it is unlikely that each stage has to learn the same amount of information, especially since each stage is initialized with the weights of the previous stage. It should therefore, in theory, be possible to train each stage only for as long as necessary, thereby potentially reducing the training time even more. We tried this

by measuring how much the generated image changes during the training of each stage (similar to what Karras et al. [2020b] did in Fig. 8) and to stop training a given stage when it does not change the output of a given image much compared to previous iterations. Again, this approach sometimes led to good results with reduced training iterations, but the results were inconsistent. However, we believe that this is a worthwhile direction and better approaches might make it possible to achieve good results with considerably fewer iterations on some (all) stages of training.

Further Improve Global Image Layout We also tried several other approaches to further improve the global image layout, especially from complex images. One idea is to add a second task for the discriminator, so that it not only has to decide whether a given image patch is real or not, but also where in a given image the patch is located (roughly). We implemented this by adding a "location loss" to the discriminator so that it also had to predict the location of a given image patch. To prevent overfitting we split the input image into nine equal rectangles (3×3) and the discriminator had to predict (for the real image only) where a given image patch is from. The generator was then trained to fool the discriminator into both predicting that the image patches of the generated images are real and to being able to correctly predict their location, too.

Our second approach was to add a second discriminator with increased receptive field to the higher stages. The idea is that this second discriminator could then still judge the global layout (and not only texture/style) even at higher resolutions. To reduce the computational burden we did not increase the convolutional filter as such, but used dilated convolutions instead. However, training still takes longer since we have to train a second discriminator at higher stages.

Both approaches had mixed results, with the added discriminator with dilated convolutions overall performing better than the location loss. The location loss often did not clearly improve the global layout and sometimes led to reduced diversity in the generated images. On the other hand, using an additional discriminator with dilated convolutions on higher stages often did actually improve the global consistency, but on average the improvements were not big enough to warrant the extra training time. We still feel that these, or similar, avenues should be further studied, since enforcing better global layout in this manner might enable us to train on even fewer stages, thereby negating the more expensive training and possibly even speeding up the overall training time.

Data Augmentation To increase the diversity in the generated images we experimented with applying basic augmentation techniques to the original training image. At each iteration we applied simple transformations such as horizontal mirroring, taking a random crop (consisting of at least 95% of the original image), slight rotation (± 5 degrees), slight zooming, etc. However, this considerably worsened the final results in our tests, since the discriminator was apparently not able to learn a good image representation and the generated images showed several artifacts (uneven textures/surfaces, no straight lines, etc). Improving/fine-tuning the used augmentation techniques or finding a more appropriate (sub-)set

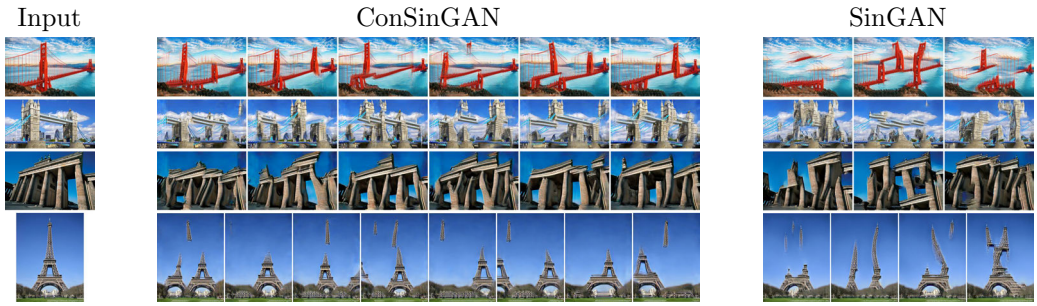


Figure B.20: Results when the models fail to learn the global image layout. The SinGAN models were trained on eight – ten stages, the ConSinGAN models were trained on three – five stages. Note that increasing the number of stages and/or the learning rate scaling δ for the ConSinGAN models improves the learned image layout.

of augmentation techniques for this task might improve results.

Different Normalization Approaches and Activation Functions We also experimented with different normalization approaches, both for the input image/noise and for the network architecture. Our generator gets as input either the original training image (normalized to a range $[-1, 1]$) for the reconstruction loss and noise sampled from a random normal distribution ($\mathcal{N}(0, 1)$) for the unconditional image generation. As such, the input to our generator comes from two slightly different distributions. We tried both normalizing the input image so that it more closely resembles a normal distribution and sampling the noise so that it follows more closely the image distribution (e.g. by sampling from $\mathcal{U}(-1, 1)$. However, both approaches did not improve the image quality or training progress and the training does, in fact, not seem to suffer from the two slightly different input distributions.

We also tried using other normalization techniques besides batch normalization in the network architecture. We tried both layer normalization [Ba et al., 2016] and pixel normalization [Karras et al., 2018], where layer normalization did not improve the results and pixel normalization made the results considerably worse. In the end we were able to completely remove any normalization layers for the unconditional image generation, which had the positive benefit of further speeding up the training. We also experimented with other activation functions besides leaky ReLU [Maas et al., 2013], such as ELU [Clevert et al., 2016], SELU [Klambauer et al., 2017], and PRELU [He et al., 2015]. PRELU usually led to similar or better results, but also slowed down training since it introduces an additional parameter. Both ELU and SELU had negative effects on the final result and in the end we kept the leaky ReLU activation function, since it works almost as well as PRELU but does not slow down the training.

B.4.7 Failure Cases

Figure B.20 shows examples of trained models that did not learn a correct global layout of the training image. Note that the SinGAN models were trained for the full eight – ten stages, while the ConSinGAN models were only trained for three – five stages. Increasing the number of trained stages to the default of six for the ConSinGAN increases the image quality.

B.4.8 Optimization and Implementation Details

Preprocessing We first rescale the input image so that its longer side has a resolution of 250 pixels for stage N and its shorter side a resolution of 25 pixels for stage 0. We then resize the origninal image to the resolutions of the intermediate stages according to Equation 3 of the original paper. The image is normalized to values in range $[-1, 1]$ during training.

Network Architecture Our discriminator and each stage of our generator consist of three convolutional layers with 64 filters each and a filter size of 3×3. Both the discriminator and the generator have two additional layers taking an image (or the noise mask) as input and extracting features, and mapping features to images (generator) or loss space (discriminator) respectively.

We use the Leaky ReLU (LReLU) activation function [Maas et al., 2013]. When BN is not used, we observe that the parameter $LReLU_\alpha$ which sets the negative slope in the LReLU does have some impact on the convergence and the final results for different tasks. For all unconditional image generation tasks we set $LReLU_\alpha = 0.05$, while setting $LReLU_\alpha = 0.3$ for other tasks such as image harmonization.

Optimization At each stage we optimize our model for 2,000 iterations (unconditional image generation) or 1,000 iterations (image harmonization and editing). We use the WGAN-GP loss [Gulrajani et al., 2017] with a gradient penalty weight of 0.1. The learning rate starts with $\eta = 0.0005$ at each stage for both the generator and the discriminator and gets multiplied with 0.1 after 80% of iterations steps at each stage. Optimization is done with Adam [Kingma and Ba, 2015] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We train three stages concurrently with a learning rate scaling of $\delta = 0.1$ for the lower stages.

During each iteration we first perform three gradient steps on the discriminator. We found that we could speed up training by only performing one gradient update on the generator during each iteration, but scaling it by a factor of 3. The scalar for the reconstruction loss in the generator is $\alpha = 10.0$. We use the leaky ReLU activation with a negative slope of 0.05 for image generation and 0.3 for image harmonization. We do not use batch normalization for unconditional image generation, but found it useful in the generator (but not the discriminator) for other tasks such as image harmonization and editing.

Training Time Training takes around 20-25 minutes for unconditional image generation and 10 minutes for image harmonization for a 250×250 pixel image on an NVIDIA GeForce GTX 1080Ti compared to training a SinGAN model which

takes on average 120-140 minutes on the same hardware. One of the reasons for this is that we only need to train our model on 5-6 stages for good results, while the original SinGAN needs 8-10 stages. However, even when both models are trained on the same number of stages we observe that it still takes less time to train our model, possibly because we do not generate an image after each stage (as SinGAN does) and do not rely as much on BN.

B.5 Learning Representations for a Single Object from Few Examples

This section gives more details about our experiments outlined in Section 5.3.

B.5.1 Keypoint Modification

Figure 5.11 shows more examples of keypoint modifications with CharacterGAN. Figure B.22 and Figure B.23 show images generated based on linearly interpolated keypoint positions between a start and end-frame, generated by our model, SinGAN [Shaham et al., 2019], ConSinGAN [Hinz et al., 2021b], and DeepSIM [Vinker et al., 2020].

B.5.2 Mask Connectivity

Figure 5.15 shows how the final image quality can be improved by ensuring that the generated mask is connected.

B.5.3 Discrete States

Figure 5.18 shows how CharacterGAN switches between discrete states based on keypoint locations.

B.5.4 Data and Reconstructions

Figure B.26 and Figure B.27 show the reconstructions of our CharacterGAN and its ablations for all training images we used for the quantitative evaluation. Figure B.28, Figure B.29, and Figure B.30 show the reconstructions of all baseline models for all training images we used for the quantitative evaluation.

B.5.5 Implementation Details

Our model is based on pix2pixHD architecture [Wang et al., 2018a], with 32 convolutional filters in the first layer of the generator and 64 convolutional filters in the first layer of the discriminator. We train our models on images of resolution 250×250 pixels. Our batch size is 5 and we train for 16,000 iterations, which takes about 60-80 minutes on an NVIDIA RTX 2080Ti. We use the Adam optimizer [Kingma and Ba, 2015] and a learning rate of 0.0002 (beta1 = 0.5) for the generator and discriminator which we linearly reduce during the last 8,000 iterations. We use instance norm [Ulyanov et al., 2016] in both the discriminator and generator. The generator uses rectified linear units (ReLU) as non-linearity, while the discriminator uses leaky ReLU with a negative slope of 0.2.

Our generator takes as input the conditioning information (250×250 pixels) and applies several convolutional layers with stride 2 until we reach a resolution of 16×16 with $1,024$ channels. We then apply nine residual blocks, each with 512

filters to this, before using transposed convolutional layers to upsample the data to the original resolution of 250×250 pixels. Our adaptive normalization technique is similar to SPADE [Park et al., 2019], however, we use different conditioning layers for each of the keypoint layers. We use two patch discriminators, one operating on the original input (250×250 pixels) and one operating on a downsampled version of the input (125×125 pixels). Both discriminators consist of five convolutional layers, of which the first three have a stride of two and the final two a stride of one.

Figure B.21: Examples from our model which was trained on only 12 – 18 images for each of the characters. Even columns show the original image and our intended modifications, odd columns show the output of our model.

Figure B.22: We show interpolated frames generated by our baselines and CharacterGAN between a *single* start and end frame (left and right columns) whose keypoint layouts are contained in the train set. All intermediate image are generated from linear interpolations of the start and end keypoint layouts and are not contained in the train set. For better comparison with the baselines we do not use the patch-based refinement step on our model for these visualization.

Figure B.23: We show interpolated frames generated by our baselines and CharacterGAN between a *single* start and end frame (left and right columns) whose keypoint layouts are contained in the train set. All intermediate image are generated from linear interpolations of the start and end keypoint layouts and are not contained in the train set. For better comparison with the baselines we do not use the patch-based refinement step on our model for these visualization.

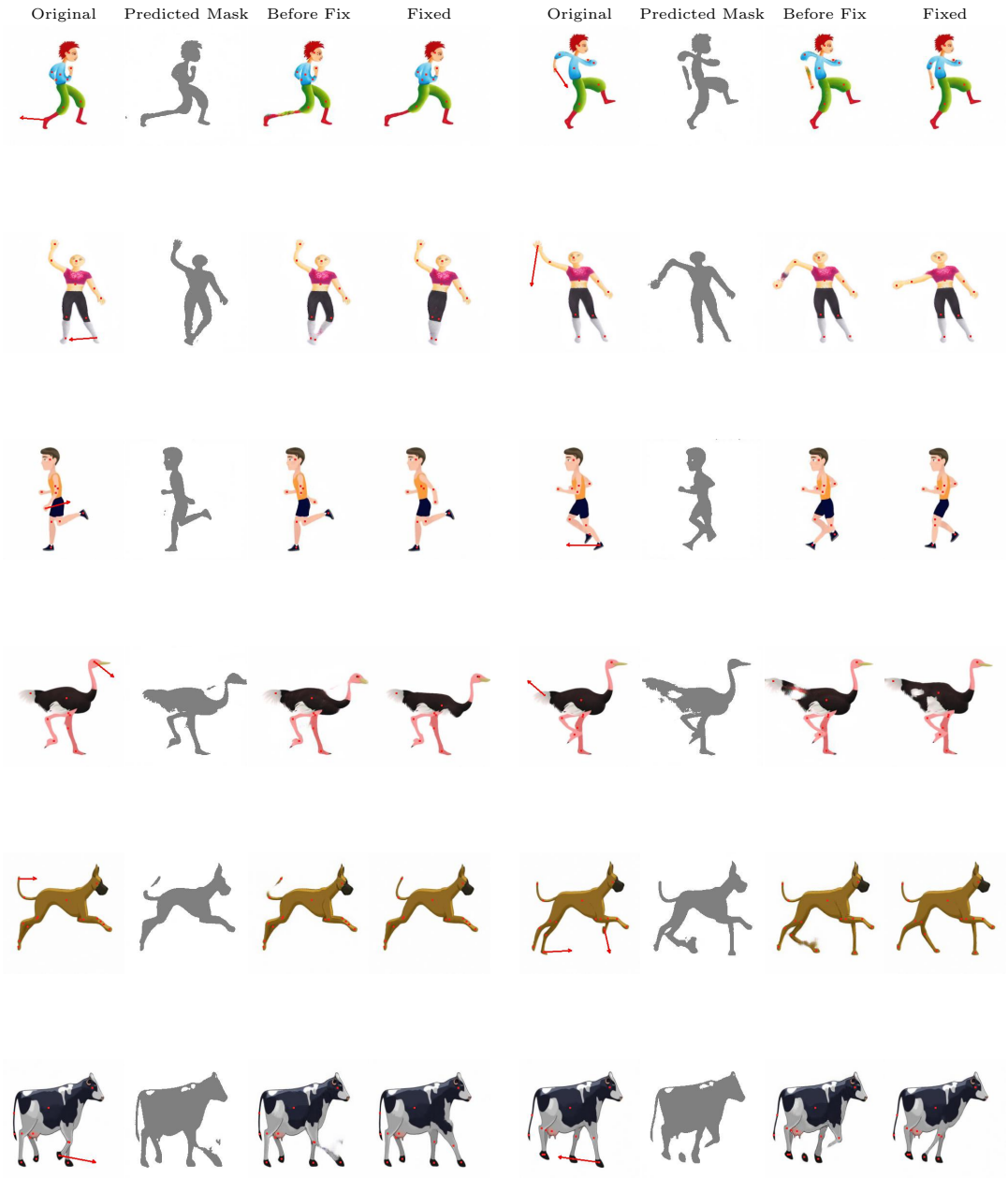


Figure B.24: Enforcing mask connectivity at test time results in more realistic images.

Figure B.25: Discrete states based on keypoint location. The first column shows the original image and intended modifications. The rest of the columns show the generated images, based on linear keypoint interpolations between the start image and the final keypoints locations based on the intended modifications.

Figure B.26: Qualitative examples of reconstructing held-out test images based on their keypoint locations.

Figure B.27: Qualitative examples of reconstructing held-out test images based on their keypoint locations.

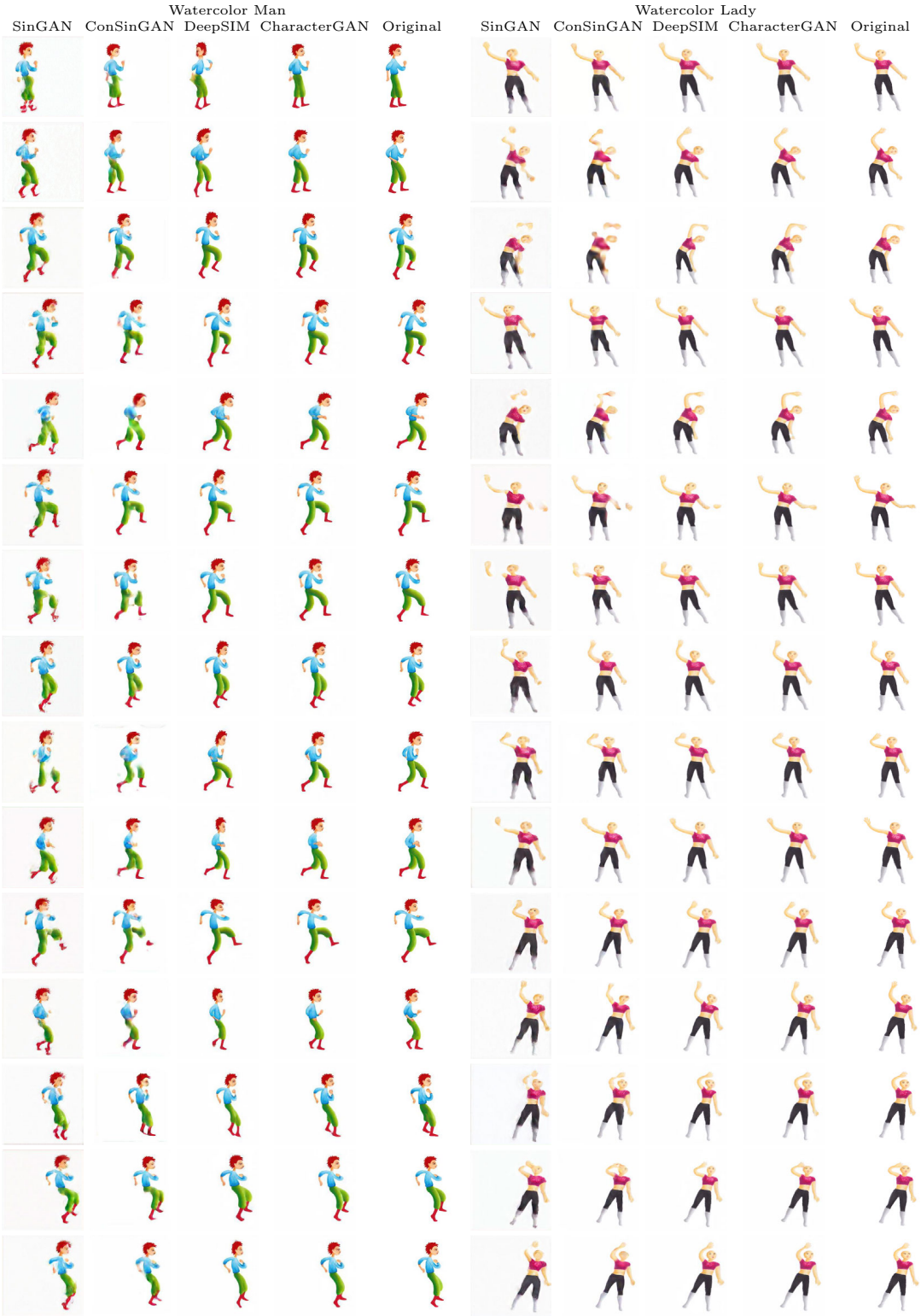


Figure B.28: Qualitative examples of reconstructing held-out test images based on their keypoint locations.

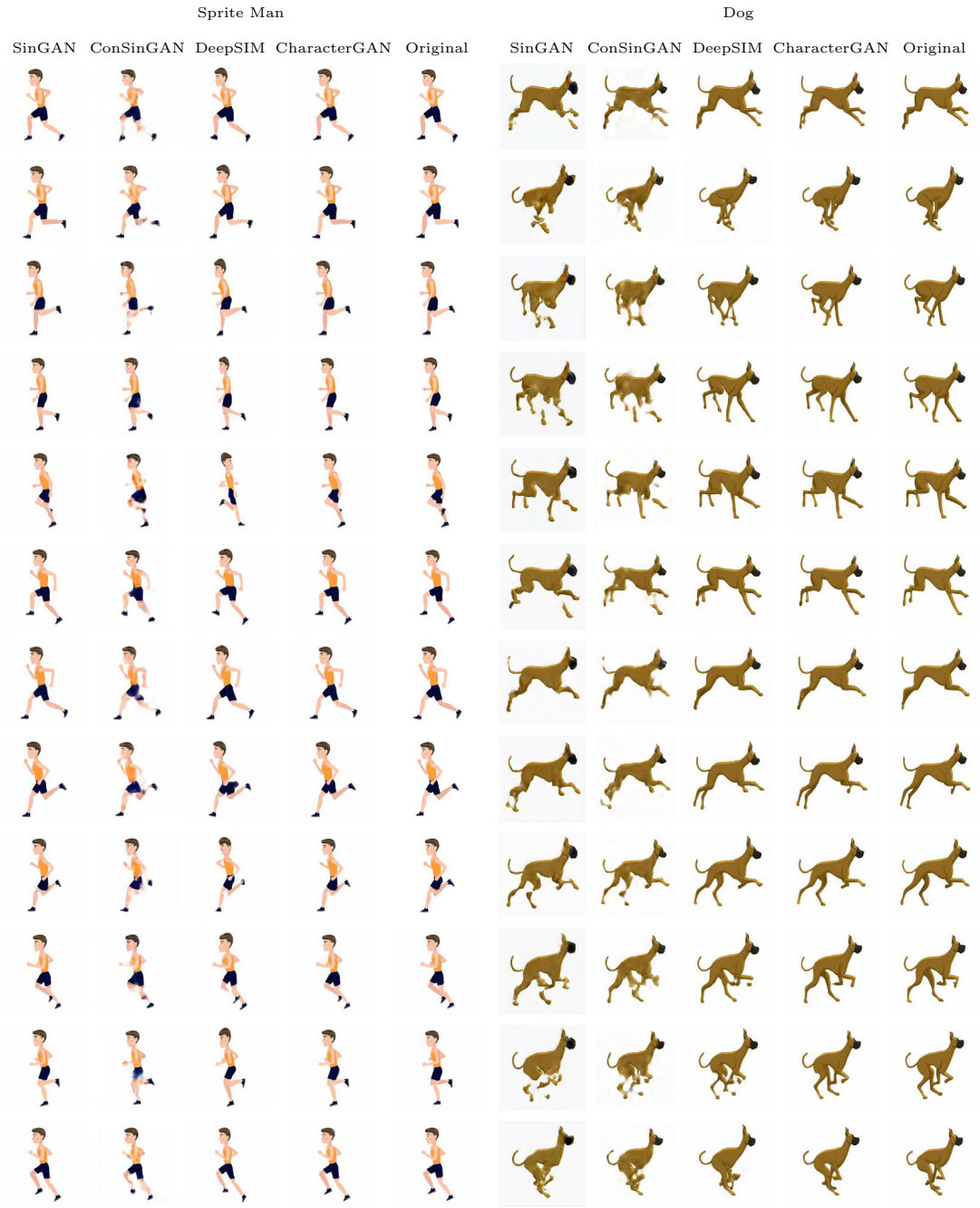


Figure B.29: Qualitative examples of reconstructing held-out test images based on their keypoint locations.

Figure B.30: Qualitative examples of reconstructing held-out test images based on their keypoint locations.

Appendix C

Acknowledgments

Doing a PhD is a long journey with many challenges and setbacks. However, few things feel better than an experiment that finally works, receiving encouraging feedback for one's work or sharing new knowledge with so many people all over the world. None of this would be possible without sharing all of these experiences and ups and downs with family, friends, and colleagues around you.

First, I want to thank Stefan Wermter for taking me in as a Master's student and then advising and working with me for almost seven years. I am grateful for all the chances and opportunities you have given me, and for giving me the freedom to explore different directions and becoming the researcher I am today. A big thank you also to Sven Magg for always being there with his advice during my whole time at the University of Hamburg and to Stefan Heinrich for taking me under his wings as a new PhD student and showing me the "big" world of research, for proof-reading my papers, and for discussing anything related to my research and career. I also want to thank Tayfun Alpay, Pablo Barros, and Doreen Jirak for many fruitful discussions throughout the last years. Finally, I want to thank Katja Kösters for always helping me out with any paperwork or documents I had to deal with and Erik Strahl for making sure the GPU-Servers were always up and running.

None of this would have been possible without the support from my family and especially my parents. You have supported me from the very start and I would not be the person I am today without you. Thank you for the years and years of emotional (and financial) support and for always letting me make my own decisions. I also want to thank my grandparents who have provided me with their companionship here in Hamburg, as well as many meals and coffees, and even their apartment when I needed it. My brother has always been there for me when I needed someone to talk to and I always enjoyed his visits and the time we spent together, be it in Hamburg, Tübingen, or Ulm.

My wife Libby has been an integral part of this whole journey and I could not have done it without her constant support and belief in myself. She has always been on my side from the very beginning and always supported my many study trips even if it meant fewer shared holidays for the two of us. Whenever I came up with new ideas of what to do next she would be right there saying "Do it". I could not have done this without you and I am looking forward to what life holds for us and our family in the next years.

Finally, I want to thank Oskar for waiting until after my defense.

Appendix D

Bibliography

A. H. Abdi, P. Abolmaesumi, and S. Fels. A preliminary study of disentanglement with insights on the inadequacy of metrics. *arXiv preprint arXiv:1911.11791*, 2019.

K. Aberman, J. Liao, M. Shi, D. Lischinski, B. Chen, and D. Cohen-Or. Neural best-buddies: Sparse cross-domain correspondence. *ACM Transactions on Graphics (TOG)*, 37(4):69, 2018.

A. Achille and S. Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

P. Agarwal, A. Betancourt, V. Panagiotou, and N. Díaz-Rodríguez. Egoshots, an ego-vision life-logging dataset and semantic fidelity metric to evaluate diversity in image captioning models. In *International Conference on Learning Representations Workshop*, 2020.

P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer, 2016.

J. Andreas. Measuring compositionality in representation learning. In *International Conference on Learning Representations*, 2019.

J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *IEEE Computer Vision and Pattern Recognition*, pages 39–48, 2016.

M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

S. Arora and Y. Zhang. Do gans actually learn the distribution? an empirical study. In *International Conference on Learning Representations*, 2018.

Y. M. Asano, C. Rupprecht, and A. Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. In *International Conference on Learning Representations*, 2020.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

P. Bachman, O. Alsharif, and D. Precup. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, pages 3365–3373, 2014.

P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pages 15535–15545, 2019.

S. Bagon, O. Boiman, and M. Irani. What is a good image segment? a unified approach to segment extraction. In *European Conference on Computer Vision*, pages 30–44. Springer, 2008.

W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang. Depth-aware video frame interpolation. In *IEEE Computer Vision and Pattern Recognition*, pages 3703–3712, 2019.

C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 28(3):24, 2009.

S. Barratt and R. Sharma. A note on the inception score. In *International Conference on Machine Learning Workshop*, 2018.

D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba. Seeing what a gan cannot generate. In *IEEE International Conference on Computer Vision*, 2019.

S. Becker and G. E. Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

S. Bell-Kligler, A. Shocher, and M. Irani. Blind super-resolution kernel estimation using an internal-gan. In *Advances in Neural Information Processing Systems*, pages 284–293, 2019.

S. Benaim, R. Mokady, A. Bermano, D. Cohen-Or, and L. Wolf. Structural-analogy from a single image pair. *arXiv preprint arXiv:2004.02222*, 2020.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

U. Bergmann, N. Jetchev, and R. Vollgraf. Learning texture manifolds with the periodic spatial gan. In *International Conference on Machine Learning*, pages 469–477, 2017.

I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

M. Bińkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan. High fidelity speech synthesis with adversarial networks. In *International Conference on Learning Representations*, 2020.

Y. Blau and T. Michaeli. The perception-distortion tradeoff. In *IEEE Computer Vision and Pattern Recognition*, pages 6228–6237, 2018.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of Conference on Computational learning theory*, pages 92–100, 1998.

A. Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:2012.08503*, 2019.

A. Casanova, M. Drozdzal, and A. Romero-Soriano. Generating unseen complex scenes: are we there yet? *arXiv preprint arXiv:2012.04027*, 2020.

M. Cha, Y. L. Gwon, and H. Kung. Adversarial learning of semantic relevance in text to image synthesis. In *AAAI International Conference on Artificial Intelligence*, pages 3272–3279, 2019.

C. Chan, S. Ginosar, T. Zhou, and A. A. Efros. Everybody dance now. In *IEEE International Conference on Computer Vision*, pages 5933–5942, 2019.

E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. *arXiv preprint arXiv:2012.00926*, 2020.

M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations*, 2017.

Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision*, pages 1520–1529, 2017.

R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

Y. Cheng, Z. Gan, Y. Li, J. Liu, and J. Gao. Sequential attention gan for interactive image editing via dialogue. In *ACM Multimedia Conference*, 2020.

T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman. The patch transform and its applications to image editing. In *IEEE Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *IEEE Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*, 2016.

T. S. Cohen and M. Welling. Transformation properties of learned visual representations. In *International Conference on Learning Representations*, 2015.

T. Dekel, T. Michaeli, M. Irani, and W. T. Freeman. Revealing and modifying non-local variations in a single image. *ACM Transactions on Graphics (TOG)*, 34(6):1–11, 2015.

U. Demir and G. Unal. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422*, 2018.

Z. Deng, J. Chen, Y. Fu, and G. Mori. Probabilistic neural programmed networks for scene generation. In *Advances in Neural Information Processing Systems*, pages 4028–4038, 2018.

E. Denton, S. Gross, and R. Fergus. Semi-supervised learning with context-conditional generative adversarial networks. *arXiv preprint arXiv:1611.06430*, 2016.

L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations Workshop*, 2015.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

K. Do and T. Tran. Theory and evaluation metrics for learning disentangled representations. In *International Conference on Learning Representations*, 2020.

C. Donahue, J. McAuley, and M. Puckette. Adversarial audio synthesis. In *International Conference on Learning Representations*, 2019.

J. Donahue and K. Simonyan. Large scale adversarial representation learning. *Advances in Neural Information Processing Systems*, pages 10542–10552, 2019.

J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *International Conference on Learning Representations*, 2017.

V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. In *International Conference on Learning Representations*, 2017.

M. Dvorožňák, W. Li, V. G. Kim, and D. Sýkora. ToonSynth: Example-based synthesis of hand-colored cartoon animations. *ACM Transactions on Graphics*, 37(4), 2018.

C. Eastwood and C. K. Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

B. Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

S. Ehrhardt, O. Groth, A. Monszpart, M. Engelcke, I. Posner, N. Mitra, and A. Vedaldi. Relate: Physically plausible multi-object scene synthesis using structured latent spaces. *Advances in Neural Information Processing Systems*, 2020.

A. El-Nouby, S. Sharma, H. Schulz, D. Hjelm, L. E. Asri, S. E. Kahou, Y. Bengio, and G. W. Taylor. Tell, draw, and repeat: Generating and modifying images based on continual linguistic instruction. In *IEEE International Conference on Computer Vision*, 2019.

M. Engelcke, A. R. Kosiorek, O. P. Jones, and I. Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations*, 2020.

S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, k. kavukcuoglu, and G. E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016.

S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *IEEE Computer Vision and Pattern Recognition*, pages 1–8, 2007.

C. Fiorio and J. Gustedt. Two linear time union-find strategies for image processing. *Theoretical Computer Science*, 154(2):165–181, 1996.

S. Frolov, T. Hinz, F. Raue, J. Hees, and A. Dengel. Adversarial text-to-image synthesis: A review. *arXiv preprint arXiv:2101.09983*, 2021.

Y. Gandelsman, A. Shocher, and M. Irani. Double-dip": Unsupervised image decomposition via coupled deep-image-priors. In *IEEE Computer Vision and Pattern Recognition*, volume 6, page 2, 2019.

R. Gao, D. Jayaraman, and K. Grauman. Object-centric representation learning from unlabeled videos. In *Asian Conference on Computer Vision*, pages 248–263, 2016.

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

J. Giménez and L. Màrquez. Linguistic features for automatic evaluation of heterogenous mt systems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics Workshop*, pages 256–264. Association for Computational Linguistics, 2007.

D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *IEEE International Conference on Computer Vision*, pages 349–356. IEEE, 2009.

I. Goodfellow. Tutorial: Generative adversarial networks. In *Advances in Neural Information Processing Systems Workshop*, 2016.

I. Goodfellow, H. Lee, Q. Le, A. Saxe, and A. Ng. Measuring invariances in deep networks. In *Advances in Neural Information Processing Systems*, pages 646–654, 2009.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, pages 529–536, 2005.

K. Greff, R. L. Kaufman, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433, 2019.

K. Greff, S. Van Steenkiste, and J. Schmidhuber. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

M. Guo, A. Fathi, J. Wu, and T. Funkhouser. Object-centric neural scene rendering. *arXiv preprint arXiv:1901.11390*, 2020.

K. He and J. Sun. Statistics of patch offsets for image completion. In *European Conference on Computer Vision*, pages 16–29. Springer, 2012.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Computer Vision and Pattern Recognition*, pages 770–778, 2016.

A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

G. E. Hinton and R. Zemel. Autoencoders, minimum description length and helmholtz free energy. *Advances in Neural Information Processing Systems*, pages 3–10, 1993.

T. Hinz and S. Wermter. Image generation and translation with disentangled representations. In *IEEE International Joint Conference on Neural Networks*, pages 5519–5526, 2018a.

T. Hinz and S. Wermter. Inferencing based on unsupervised learning of disentangled representations. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 61–66, 2018b.

T. Hinz, S. Heinrich, and S. Wermter. Generating multiple objects at spatially distinct locations. In *International Conference on Learning Representations*, 2019.

T. Hinz, S. Heinrich, and S. Wermter. Semantic object accuracy for generative text-to-image synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):1–14, 2020.

T. Hinz, M. Fisher, O. Wang, E. Shechtman, and S. Wermter. Charactergan: Few-shot keypoint character animation and reposing. *arXiv preprint arXiv:2102.03141*, 2021a.

T. Hinz, M. Fisher, O. Wang, and S. Wermter. Improved techniques for training single-image gans. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1300–1309, 2021b.

R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18(1-3):65–96, 1984.

S. Hong, X. Yan, T. Huang, and H. Lee. Learning hierarchical semantic image manipulation through structured representations. In *Advances in Neural Information Processing Systems*, pages 2712–2722, 2018a.

S. Hong, D. Yang, J. Choi, and H. Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *IEEE Computer Vision and Pattern Recognition*, pages 7986–7994, 2018b.

J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *IEEE Computer Vision and Pattern Recognition*, pages 5197–5206, 2015.

W. Huang, R. Y. D. Xu, and I. Oppermann. Realistic image generation using region-phrase attention. In *Asian Conference on Machine Learning*, pages 284–299, 2019a.

X. Huang, M. Wang, and M. Gong. Hierarchically-fused generative adversarial network for text to realistic image synthesis. In *IEEE Conference on Computer and Robot Vision*, pages 73–80, 2019b.

J. Hupé, A. James, B. Payne, S. Lomber, P. Girard, and J. Bullier. Cortical feedback improves discrimination between figure and background by v1, v2 and v3 neurons. *Nature*, 394(6695):784–787, 1998.

P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

O. Jamriska. Ebsynth: Fast example-based image synthesis and style transfer. https://github.com/jamriska/ebsynth, 2018.

N. Jetchev, U. Bergmann, and R. Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016.

J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *IEEE Computer Vision and Pattern Recognition*, pages 1988–1997, 2017.

J. Johnson, A. Gupta, and L. Fei-Fei. Image generation from scene graphs. In *IEEE Computer Vision and Pattern Recognition*, pages 1219–1228, 2018.

A. A. Jyothi, T. Durand, J. He, L. Sigal, and G. Mori. Layoutvae: Stochastic scene layout generation from a label set. *IEEE International Conference on Computer Vision*, 2019.

M. Kang and J. Park. Contragan: Contrastive learning for conditional image generation. *Advances in Neural Information Processing Systems*, 2020.

L. Karacan, Z. Akata, A. Erdem, and E. Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv preprint arXiv:1612.00215*, 2016.

A. Karnewar and O. Wang. Msg-gan: Multi-scale gradients for generative adversarial networks. In *IEEE Computer Vision and Pattern Recognition*, 2020.

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems*, 2020a.

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *IEEE Computer Vision and Pattern Recognition*, pages 8110–8119, 2020b.

M. Kilickaya, A. Erdem, N. Ikizler-Cinbis, and E. Erdem. Re-evaluating automatic metrics for image captioning. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 199–209, 2017.

H. Kim and A. Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658, 2018.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980, 2017.

M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

A. Kumar, P. Sattigeri, and A. Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2018.

B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, et al. Fader networks: Generating image variations by sliding attribute values. In *Advances in Neural Information Processing Systems*, pages 5963–5972, 2017.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

D. Lee, M.-Y. Liu, M.-H. Yang, S. Liu, J. Gu, and J. Kautz. Context-aware synthesis and placement of object instances. In *Advances in Neural Information Processing Systems*, pages 10413–10423, 2018.

K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *IEEE Computer Vision and Pattern Recognition*, pages 991–999, 2015.

B. Li, X. Qi, T. Lukasiewicz, and P. Torr. Controllable text-to-image generation. In *Advances in Neural Information Processing Systems*, pages 2065–2075, 2019a.

B. Li, B. Zhuang, M. Li, and J. Gu. Seq-sg2sl: Inferring semantic layout from scene graph through sequence to sequence learning. In *IEEE International Conference on Computer Vision*, 2019b.

C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

C. Li, H. Liu, C. Chen, Y. Pu, L. Chen, R. Henao, and L. Carin. Alice: Towards understanding adversarial learning for joint distribution matching. In *Advances in Neural Information Processing Systems*, pages 5495–5503, 2017a.

C. Li, T. Xu, J. Zhu, and B. Zhang. Triple generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 4091–4101, 2017b.

J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu. Layoutgan: Generating graphic layouts with wireframe discriminators. *International Conference on Learning Representations*, 2019c.

N. Li, R. Fisher, et al. Learning object-centric representations of multi-object scenes from multiple views. *Advances in Neural Information Processing Systems*, 2020a.

W. Li, P. Zhang, L. Zhang, Q. Huang, X. He, S. Lyu, and J. Gao. Object-driven text-to-image synthesis via adversarial training. In *IEEE Computer Vision and Pattern Recognition*, pages 12174–12182, 2019d.

Y. Li, Z. Gan, Y. Shen, J. Liu, Y. Cheng, Y. Wu, L. Carin, D. Carlson, and J. Gao. Storygan: A sequential conditional gan for story visualization. In *IEEE Computer Vision and Pattern Recognition*, pages 6329–6338, 2019e.

Y. Li, T. Ma, Y. Bai, N. Duan, S. Wei, and X. Wang. Pastegan: A semi-parametric method to generate image from scene graph. In *Advances in Neural Information Processing Systems*, 2019f.

Y. Li, R. Zhang, J. C. Lu, and E. Shechtman. Few-shot image generation with elastic weight consolidation. *Advances in Neural Information Processing Systems*, 2020b.

Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020c.

J. Lin, Y. Pang, Y. Xia, Z. Chen, and J. Luo. Tuigan: Learning versatile image-to-image translation with two unpaired images. *European Conference on Computer Vision*, 2020.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

S. Liu, A. Jacobson, and Y. Gingold. Skinning cubic bézier splines and catmull-clark subdivision surfaces. *ACM Transactions on Graphics (TOG)*, 33(6):1–9, 2014.

X. Liu, G. Yin, J. Shao, X. Wang, et al. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *Advances in Neural Information Processing Systems*, pages 568–578, 2019.

Z. Liu, P. Luo, W. Wang, and X. Tang. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

F. Locatello, G. Abbati, T. Rainforth, S. Bauer, B. Schölkopf, and O. Bachem. On the fairness of disentangled representations. In *Advances in Neural Information Processing Systems*, pages 14611–14624, 2019a.

F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pages 4114–4124, 2019b.

F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. A commentary on the unsupervised learning of disentangled representations. In *AAAI International Conference on Artificial Intelligence*, volume 34, pages 13681–13684, 2020a.

F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. A sober look at the unsupervised learning of disentangled representations and their evaluation. *Journal of Machine Learning Research*, 21:1–62, 2020b.

F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 2020c.

H.-M. Lu, Y. Fainman, and R. Hecht-Nielsen. Image manifolds. In *Applications of Artificial Neural Networks in Image Processing III*, volume 3307, pages 52–63, 1998.

F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep painterly harmonization. *Computer Graphics Forum*, 37(4):95–106, 2018.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, volume 30, 2013.

P. S. Madhyastha, J. Wang, and L. Specia. Vifidel: Evaluating the visual fidelity of image descriptions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 6539–6550, 2019.

E. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov. Generating images from captions with attention. In *International Conference on Learning Representations*, 2016.

J. Mao, X. Zhang, Y. Li, W. T. Freeman, J. B. Tenenbaum, and J. Wu. Program-guided image manipulators. In *IEEE International Conference on Computer Vision*, pages 4030–4039, 2019a.

Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *IEEE Computer Vision and Pattern Recognition*, pages 1429–1437, 2019b.

I. D. Mastan and S. Raman. Multi-level encoder-decoder architectures for image restoration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

I. D. Mastan and S. Raman. Dcil: Deep contextual internal learning for image restoration and image retargeting. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2366–2375, 2020.

R. Mechrez, E. Shechtman, and L. Zelnik-Manor. Saliency driven image manipulation. *Machine Vision and Applications*, 30(2):189–202, 2019.

L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning*, pages 3481–3490, 2018.

L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *International Conference on Learning Representations*, 2017.

T. Michaeli and M. Irani. Blind deblurring using internal patch recurrence. In *European Conference on Computer Vision*, pages 783–798. Springer, 2014.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, 2020.

M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

G. Mittal, S. Agrawal, A. Agarwal, S. Mehta, and T. Marwah. Interactive image generation using scene graphs. In *International Conference on Learning Representations Workshop*, 2019.

T. Miyato and M. Koyama. cgans with projection discriminator. In *International Conference on Learning Representations*, 2018.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

S. Mo, M. Cho, and J. Shin. Freeze discriminator: A simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*, 2020.

H. Mobahi, C. Liu, and W. T. Freeman. A compositional model for low-dimensional image set representation. In *IEEE Computer Vision and Pattern Recognition*, pages 1322–1329, 2014.

A. S. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018.

Y. Mroueh, T. Sercu, and V. Goel. Mcgan: Mean and covariance feature matching gan. In *International Conference on Machine Learning*, pages 2527–2535, 2017.

S. Nam, Y. Kim, and S. J. Kim. Text-adaptive generative adversarial networks: manipulating images with natural language. In *Advances in Neural Information Processing Systems*, pages 42–51, 2018.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems Workshop*, 2011.

A. Nguyen, Y. Bengio, and A. Dosovitskiy. Plug & play generative networks: Conditional iterative generation of images in latent space. In *IEEE Computer Vision and Pattern Recognition*, pages 4467–4477, 2017.

T. Nguyen-Phuoc, C. Richardt, L. Mai, Y.-L. Yang, and N. Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. *Advances in Neural Information Processing Systems*, 2020.

M. Niemeyer and A. Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. *arXiv preprint arXiv:2011.12100*, 2020.

A. Noguchi and T. Harada. Image generation from small datasets via batch statistics adaptation. In *IEEE International Conference on Computer Vision*, pages 2750–2758, 2019.

A. Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *International Conference on Machine Learning*, pages 2642–2651, 2017.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

B. Paige, J.-W. van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, P. Torr, et al. Learning disentangled representations with semi-supervised deep generative models. *Advances in Neural Information Processing Systems*, pages 5925–5935, 2017.

T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, 2020.

G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez. Invertible Conditional GANs for image editing. In *Advances in Neural Information Processing Systems Workshop*, 2016.

J. Pont-Tuset, J. Uijlings, S. Changpinyo, R. Soricut, and V. Ferrari. Connecting vision and language with localized narratives. In *European Conference on Computer Vision*, pages 647–664, 2020.

O. Poursaeed, V. Kim, E. Shechtman, J. Saito, and S. Belongie. Neural puppet: Generative layered cartoon characters. In *IEEE Winter Conference on Applications of Computer Vision*, pages 3346–3356, 2020.

T. Qiao, J. Zhang, D. Xu, and D. Tao. Learn, imagine and create: Text-to-image generation from prior knowledge. In *Advances in Neural Information Processing Systems*, pages 885–895, 2019a.

T. Qiao, J. Zhang, D. Xu, and D. Tao. Mirrorgan: Learning text-to-image generation by redescription. In *IEEE Computer Vision and Pattern Recognition*, pages 1505–1514, 2019b.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

A. Raj, C. Ham, H. Alamri, V. Cartillier, S. Lee, and J. Hays. Compositional generation of images. In *Advances in Neural Information Processing Systems Workshop*, 2017.

S. Ravuri and O. Vinyals. Classification accuracy score for conditional generative models. In *Advances in Neural Information Processing Systems*, 2019.

J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. Learning what and where to draw. In *Advances in Neural Information Processing Systems*, pages 217–225, 2016a.

S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069, 2016b.

S. Reed, A. van den Oord, N. Kalchbrenner, V. Bapst, M. Botvinick, and N. de Freitas. Generating interpretable images with controllable structure. 2016c.

E. Robb, W.-S. Chu, A. Kumar, and J.-B. Huang. Few-shot adaptation of generative adversarial networks. *arXiv preprint arXiv:2010.11943*, 2020.

C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *IEEE Winter Conference on Applications of Computer Vision*, pages 29–36, 2005.

K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems*, pages 2018–2028, 2017.

S. Sah, D. Peri, A. Shringi, C. Zhang, M. Dominguez, A. Savakis, and R. Ptucha. Semantically invariant text-to-image generation. In *IEEE International Conference on Image Processing*, pages 3783–3787, 2018.

M. Saito, E. Matsumoto, and S. Saito. Temporal generative adversarial nets with singular value clipping. In *IEEE Computer Vision and Pattern Recognition*, pages 2830–2839, 2017.

M. Saito, S. Saito, M. Koyama, and S. Kobayash. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal gan. *International Journal of Computer Vision*, 128:2586–2606, 2020.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 2020.

A. Sepliarskaia, J. Kiseleva, and M. de Rijke. Evaluating disentangled representations. *arXiv preprint arXiv:1910.05587*, 2019.

T. R. Shaham, T. Dekel, and T. Michaeli. Singan: Learning a generative model from a single natural image. In *IEEE International Conference on Computer Vision*, pages 4570–4580, 2019.

S. Sharma, D. Suhubdy, V. Michalski, S. E. Kahou, and Y. Bengio. Chatpainter: Improving text to image generation using dialogue. In *International Conference on Learning Representations Workshop*, 2018.

W. Shen and R. Liu. Learning residual images for face attribute manipulation. In *IEEE Computer Vision and Pattern Recognition*, pages 1225–1233, 2017.

K. Shmelkov, C. Schmid, and K. Alahari. How good is my gan? In *European Conference on Computer Vision*, pages 213–229, 2018.

A. Shocher, N. Cohen, and M. Irani. "zero-shot" super-resolution using deep internal learning. In *IEEE Computer Vision and Pattern Recognition*, pages 3118–3126, 2018.

A. Shocher, S. Bagon, P. Isola, and M. Irani. Ingan: Capturing and retargeting the dna of a natural image. In *IEEE International Conference on Computer Vision*, pages 4492–4501, 2019.

A. Siarohin, S. Lathuilière, S. Tulyakov, E. Ricci, and N. Sebe. First order motion model for image animation. In *Advances in Neural Information Processing Systems*, 2019.

D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *IEEE Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

S. Sinha, Z. Zhao, A. G. Alias Parth Goyal, C. A. Raffel, and A. Odena. Top-k training of gans: Improving gan performance by throwing away bad samples. *Advances in Neural Information Processing Systems*, 2020.

V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *IEEE Computer Vision and Pattern Recognition*, pages 2437–2446, 2019a.

V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1121–1132, 2019b.

V. Sitzmann, E. Chan, R. Tucker, N. Snavely, and G. Wetzstein. Metasdf: Meta-learning signed distance functions. *Advances in Neural Information Processing Systems*, 2020a.

V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 2020b.

E. S. Spelke. Principles of object perception. *Cognitive science*, 14(1):29–56, 1990.

A. Spurr, E. Aksan, and O. Hilliges. Guiding infogan with semi-supervision. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 119–134, 2017.

A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *Advances in neural information processing systems*, 30:3308–3318, 2017.

A. Stone, H. Wang, M. Stark, Y. Liu, D. Scott Phoenix, and D. George. Teaching compositionality to cnns. In *IEEE Computer Vision and Pattern Recognition*, pages 5058–5067, 2017.

W. Sun and T. Wu. Image synthesis from reconfigurable layout and style. In *IEEE International Conference on Computer Vision*, 2019.

R. Suter, D. Miladinovic, B. Schölkopf, and S. Bauer. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *International Conference on Machine Learning*, pages 6056–6065, 2019.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

D. Tabernik, M. Kristan, J. L. Wyatt, and A. Leonardis. Towards deep compositional networks. In *International Conference on Pattern Recognition*, pages 3470–3475, 2016.

F. Tan, S. Feng, and V. Ordonez. Text2scene: Generating abstract scenes from textual descriptions. *IEEE Computer Vision and Pattern Recognition*, 2019.

M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. T. Barron, and R. Ng. Learned initializations for optimizing coordinate-based neural representations. *arXiv preprint arXiv:2012.02189*, 2020.

W. Tang, P. Yu, J. Zhou, and Y. Wu. Towards a unified compositional model for visual pattern modeling. In *IEEE International Conference on Computer Vision*, pages 2784–2793, 2017.

O. Texler, D. Futschik, J. Fišer, M. Lukáč, J. Lu, E. Shechtman, and D. Sỳkora. Arbitrary style transfer using neurally-guided patch-based synthesis. *Computers & Graphics*, 87:62–71, 2020.

L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, 2016.

N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop*, pages 1–5, 2015.

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the Annual Allerton Conference on Communications, Control and Computing*, pages 368–377, 1999.

T. Tlusty, T. Michaeli, T. Dekel, and L. Zelnik-Manor. Modifying non-local variations across multiple views. In *IEEE Computer Vision and Pattern Recognition*, pages 6276–6285, 2018.

P. Tokmakov, Y.-X. Wang, and M. Hebert. Learning compositional representations for few-shot recognition. In *IEEE International Conference on Computer Vision*, pages 6372–6381, 2019.

N.-T. Tran, V.-H. Tran, N.-B. Nguyen, T.-K. Nguyen, and N.-M. Cheung. Towards good practices for data augmentation in gan training. *arXiv preprint arXiv:2006.05338*, 2020.

F. Träuble, E. Creager, N. Kilbertus, A. Goyal, F. Locatello, B. Schölkopf, and S. Bauer. Is independence all you need? on the generalization of representations learned from correlated data. *arXiv preprint arXiv:2006.07886*, 2020.

M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic. On mutual information maximization for representation learning. In *International Conference on Learning Representations*, 2020.

P.-H. Tseng, R. Carmi, I. G. Cameron, D. P. Munoz, and L. Itti. Quantifying center bias of observers in free viewing of dynamic natural scenes. *Journal of vision*, 9(7):4–4, 2009.

S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. In *IEEE Computer Vision and Pattern Recognition*, pages 1526–1535, 2018.

D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *IEEE Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.

S. Van Steenkiste, F. Locatello, J. Schmidhuber, and O. Bachem. Are disentangled representations helpful for abstract visual reasoning? In *Advances in Neural Information Processing Systems*, pages 14245–14258, 2019.

S. Van Steenkiste, K. Kurach, J. Schmidhuber, and S. Gelly. Investigating object compositionality in generative adversarial networks. *Neural Networks*, 130:309 – 325, 2020.

N. Vasconcelos and A. Lippman. A multiresolution manifold distance for invariant image similarity. *IEEE Transactions on Multimedia*, 7(1):127–142, 2005.

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11 (12), 2010.

Y. Vinker, E. Horwitz, N. Zabari, and Y. Hoshen. Deep single image manipulation. *arXiv preprint arXiv:2007.01289*, 2020.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):652–663, 2016.

D. M. Vo and A. Sugimoto. Visual-relation conscious image generation from structured-text. In *European Conference on Computer Vision*, 2020.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

J. Wang and A. L. Yuille. Semantic part segmentation using compositional model combining shape and appearance. In *IEEE Computer Vision and Pattern Recognition*, pages 1788–1797, 2015.

T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Computer Vision and Pattern Recognition*, pages 8798–8807, 2018a.

Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu. Transferring gans: generating images from limited data. In *European Conference on Computer Vision*, pages 218–234, 2018b.

Y. Wang, A. Gonzalez-Garcia, D. Berga, L. Herranz, F. S. Khan, and J. v. d. Weijer. Minegan: effective knowledge transfer from gans to target domains with few images. In *IEEE Computer Vision and Pattern Recognition*, pages 9332–9341, 2020.

J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. In *Neural networks: Tricks of the trade*, pages 639–655. Springer, 2012.

K. Wu, E. Otoo, and A. Shoshani. Optimizing connected component labeling algorithms. In *Medical Imaging 2005: Image Processing*, volume 5747, pages 1965–1976, 2005.

K. Xu, H. Liang, J. Zhu, H. Su, and B. Zhang. Deep structured generative models. In *International Conference on Machine Learning Workshop*, 2018a.

T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *IEEE Computer Vision and Pattern Recognition*, 2018b.

W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, 2019.

G. Yin, B. Liu, L. Sheng, N. Yu, X. Wang, and J. Shao. Semantics disentangling for text-to-image generation. In *IEEE Computer Vision and Pattern Recognition*, pages 2327–2336, 2019.

F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

S. Yu, H. Dong, F. Liang, Y. Mo, C. Wu, and Y. Guo. Simgan: Photo-realistic semantic image manipulation using generative adversarial networks. In *IEEE International Conference on Image Processing*, pages 734–738, 2019.

A. Yuille and R. Mottaghi. Complexity of representation and inference in compositional models with part sharing. *The Journal of Machine Learning Research*, 17 (1):292–319, 2016.

H. Zhang, Z. Deng, X. Liang, J. Zhu, and E. Xing. Structured generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 3900–3910, 2017a.

H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE International Conference on Computer Vision*, pages 5907–5915, 2017b.

H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2018a.

H. Zhang, L. Mai, N. Xu, Z. Wang, J. Collomosse, and H. Jin. An internal learning approach to video inpainting. In *IEEE International Conference on Computer Vision*, pages 2720–2729, 2019.

H. Zhang, J. Yu Koh, J. Baldridge, H. Lee, and Y. Yang. Cross-modal contrastive learning for text-to-image generation. *arXiv preprint arXiv:2101.04702*, 2021.

K. Zhang, G. Riegler, N. Snavely, and V. Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Computer Vision and Pattern Recognition*, pages 586–595, 2018b.

S. Zhang, H. Dong, W. Hu, Y. Guo, C. Wu, D. Xie, and F. Wu. Text-to-image synthesis via visual-memory creative adversarial network. In *Pacific Rim Conference on Multimedia*, pages 417–427, 2018c.

Z. Zhang, Y. Xie, and L. Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In *IEEE Computer Vision and Pattern Recognition*, pages 6199–6208, 2018d.

B. Zhao, L. Meng, W. Yin, and L. Sigal. Image generation from layout. In *IEEE Computer Vision and Pattern Recognition*, pages 8584–8593, 2019.

M. Zhao, Y. Cong, and L. Carin. On leveraging pretrained gans for generation with limited data. In *International Conference on Machine Learning*, 2020a.

S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. Differentiable augmentation for data-efficient gan training. In *Advances in Neural Information Processing Systems*, 2020b.

Z. Zhao, Z. Zhang, T. Chen, S. Singh, and H. Zhang. Image augmentations for gan training. *arXiv preprint arXiv:2006.02595*, 2020c.

B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014.

X. Zhou, S. Huang, B. Li, Y. Li, J. Li, and Z. Zhang. Text guided person image synthesis. In *IEEE Computer Vision and Pattern Recognition*, pages 3663–3672, 2019.

Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

Z.-H. Zhou and M. Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541, 2005.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, pages 2223–2232, 2017a.

J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017b.

M. Zhu, P. Pan, W. Chen, and Y. Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *IEEE Computer Vision and Pattern Recognition*, pages 5802–5810, 2019.

X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.

M. Zontak and M. Irani. Internal statistics of a single natural image. In *IEEE Computer Vision and Pattern Recognition*, pages 977–984. IEEE, 2011.

M. Zontak, I. Mosseri, and M. Irani. Separating signal from noise using patch recurrence across scales. In *IEEE Computer Vision and Pattern Recognition*, pages 1195–1202, 2013.

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, 8. Februar 2021

Ort, Datum

Unterschrift

Erklärung zur Veröffentlichung

Ich stimme der Einstellung der Dissertation in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, 8. Februar 2021

Ort, Datum

Unterschrift