Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Challenges and Solutions for the Protection of Training Data of Machine Learning Models

CUMULATIVE DISSERTATION

with the aim of achieving a doctoral degree at the

Faculty of Mathematics, Informatics and Natural Sciences

Department of Informatics

University of Hamburg

submitted by

Joshua Lukas Stock

March 30, 2025

Reviewers:

Prof. Dr. Hannes Federrath
Prof. Dr. Konrad Rieck

Date of the oral defense:

July 9, 2025

Abstract

Few technologies have had as great an impact on society in recent years as machine learning (ML) algorithms, and their influence continues to grow. Many applications, such as voice assistants, self-driving vehicles, and advanced chatbots, would not be possible without ML models at their core. Before being deployed, these models go through a training phase where they are optimized using a training dataset. This automated process is the key to the popularity of ML models: Models can answer complex questions represented by data in different forms – including image, audio and tabular data. Successful model training often requires vast amounts of training data. The increasing digitization of business processes has led many organizations to collect and store data on large scales, which can be used for ML training. However, this data may contain sensitive information, such as personal information or trade secrets. Using it for training and publishing a trained model afterwards does not come without risks.

As numerous publications have shown, ML models often store more sensitive information of their training data than intended, even if this is not necessary for the fulfillment of their tasks. In particular, so-called inference attacks aim at recovering such sensitive information from models *after* their training process. This dissertation fills a research gap regarding the reconstruction of statistical training data properties from ML models – the property inference attack (PIA). Both the white-box threat model, where the attacker has access to the model's internal parameters, and the black-box threat model, where the attacker can only compute model outputs through an interface, are examined. For the white-box variant, a defense mechanism is introduced and evaluated. Outlining its limitations, the functionality of the attack is analyzed, revealing severe traces of training data properties in the trained parameters of ML models. For the black-box scenario of the PIA, a new attack framed as a regression problem is proposed. Its performance is tested in experiments and compared to a white-box benchmark, exhibiting strong R^2 test values of up to 0.86. To defend against this attack, a promising adversarial learning defense strategy is presented and experimentally evaluated.

When training data is spread across multiple parties, distributed training algorithms such as federated learning (FL) enable the collaborative training of ML models without the transmission of training data. This dissertation examines the benefits and practical limitations of FL through a case study in official statistics. Simulations across three use cases – medical insurance, fine dust pollution, and mobile radio coverage – demonstrate its strong potential for producing official statistics. FL is particularly advantageous when data owners wish to retain control over their data while still benefiting from a collaboratively trained model. A common privacy concern in FL is the potential leakage of sensitive information through data exchanged during the training process. To prevent leaked information from being linked to individual participants, a novel FL protocol that enhances client anonymity is also presented in this dissertation. An independent dealer party is introduced to facilitate an efficient cryptographic masking mechanism, reducing runtimes by up to 87.8% compared to related work. The security of the protocol is validated by a mathematical proof and its performance is assessed in various experiments.

As ML algorithms continue to gain relevance, this dissertation aims to contribute to an understanding and improvement of training data privacy, both for PIAs on trained models and distributed FL training processes.

Zusammenfassung

Die gesellschaftliche Relevanz von Algorithmen des maschinellen Lernens (ML) ist in den letzten Jahren stetig gestiegen. Moderne Anwendungen wie Sprachassistenten, selbstfahrende Fahrzeuge und Chatbots basieren zunehmend auf leistungsfähigen ML-Modellen. Bevor solche Modelle eingesetzt werden können, durchlaufen sie eine Trainingsphase, in der sie schrittweise an einen Trainingsdatensatz angepasst bzw. dafür optimiert werden. Dieser algorithmengesteuerte Anpassungsprozess ist von entscheidender Bedeutung für den Erfolg der Technologie: Er ermöglicht es den ML-Modellen, komplexe Problemlösungen aus den vorliegenden Daten abzuleiten. Diese Daten können in verschiedenen Formaten vorliegen, etwa als tabellarische Daten, Bild- oder Audiodateien. Für ein erfolgreiches Training sind oftmals große Mengen an Daten erforderlich. Begünstigt durch die fortschreitende Digitalisierung fallen in vielen Bereichen enorme Mengen an Daten an, die zum Training von ML-Modellen genutzt werden können. Allerdings enthalten viele dieser Datensätze auch schützenswerte Informationen, etwa Geschäftsgeheimnisse oder personenbezogene Daten, die nicht für die Öffentlichkeit bestimmt sind. Der Einsatz solcher Datensätze zum Training eines ML Modells und dessen anschließende Veröffentlichung können Gefahren für den Datenschutz mit sich bringen.

Wie in zahlreichen wissenschaftlichen Arbeiten gezeigt werden konnte, beinhalten trainierte ML-Modelle oft mehr sensible Informationen aus den Trainingsdaten als notwendig. Sogenannte Inferenzangriffe zielen darauf ab, diese sensiblen Informationen aus den trainierten Modellen zu extrahieren. Die vorliegende Dissertation schließt eine Forschungslücke im Bereich der Rekonstruktion von statistischen Trainingsdateneigenschaften – der sogenannten *Property Inference*. Dabei werden sowohl das White-Box-Angreifermodell, bei dem Angreifende Zugriff auf die internen Modellparameter haben, als auch das Black-Box-Angreifermodell, bei dem lediglich über eine Schnittstelle Modellausgaben berechnen werden können, untersucht. Für das White-Box-Modell wird ein neuer Verteidigungsmechanismus vorgestellt und evaluiert, wobei festgestellt wird, dass seine Wirksamkeit auf bestimmte Instanzen von Property Inference Angreifern, die zum Zeitpunkt der Verteidigung bekannt sein müssen, limitiert ist. Die Analyse jener Einschränkung führt zu einer tiefergehenden Untersuchung der Funktionsweise von White-Box Property Inference. Dabei wird gezeigt, dass die Spuren statistischer Eigenschaften der Trainingsdaten über weite Teile der trainierten Modellparameter hinweg verteilt sind und häufig deutlich erkennbar sind. Im Black-Box-Angreifermodell wird eine neue Version des Angriffs vorgestellt, bei der Property Inference nicht wie herkömmlich als Klassifikations-, sondern als Regressionsproblem betrachtet wird. In eigens durchgeführten Experimenten wird die Leistungsfähigkeit dieses Angriffs mit einem R^2-Wert von bis zu 0,86 demonstriert, und mit einem entsprechenden White-Box-Angriff verglichen. Ein vielversprechender Verteidigungsmechanismus gegen diesen Black-Box-Angriff, der auf *adversarial learning* basiert, wird ebenfalls eingeführt und in Experimenten auf seine Wirksamkeit hin evaluiert.

In der Praxis sind Trainingsdaten oftmals nicht an einem zentralen Ort gebündelt, sondern auf mehrere Parteien verteilt. Verteilte Trainingsalgorithmen wie *Federated Learning* (FL) ermöglichen es, ein gemeinsames Modell zu trainieren, ohne dass die Trainingsdaten selbst übertragen werden müssen. Diese Dissertation enthält eine Analyse der Vorteile und der praktischen Herausforderungen von FL, die aus einer simulierten Fallstudie zum Potenzial von FL für die amtliche Statistik abgeleitet wird. Die Simulationen beziehen sich auf Themen, die für die amtliche Statistik von Bedeutung sind, nämlich Luftverschmutzung, Kosten von Krankenversicherungen und die Empfangsabdeckung für den Mobilfunk. Die Ergebnisse dieser Simulationen zeigen, dass FL in Bezug auf die Modellleistung mit herkömmlichen Trainingsalgorithmen konkurrieren kann. Daher bietet FL besonders in Szenarien, in denen Besitzer von Trainingsdaten ihre Datenhoheit nicht aufgeben möchten, ein großes Potenzial. Allerdings bergen auch FL-Trainingsalgorithmen Risiken für die Geheimhaltung sensibler Informationen aus den Trainingsdaten. Denn das Training durch FL beinhaltet regelmäßige Übertragungen von trainierten Modellen zwischen einer zentralen Partei und den Teilnehmern – den sogenannten *Clients*. Um zu verhindern, dass sensible Informationen einzelnen Clients zugeordnet werden können, wird in dieser Dissertation ein neuartiges Protokoll vorgestellt, das eine verbesserte Anonymität der Teilnehmer ermöglicht. Dazu wird eine unabhängige Dealer-Partei eingeführt, die rechenintensive kryptografische Operationen übernimmt und somit einen effizienten Algorithmus zur Maskierung der Client-Updates ermöglicht. Im Vergleich zu verwandten Arbeiten kann die Laufzeit des Protokolls um bis zu 87,8% reduziert werden. Experimente belegen sowohl die Effizienz als auch die Skalierbarkeit des neuen Protokolls, dessen Sicherheit zudem mathematisch bewiesen wird.

Mit der stetig wachsenden Relevanz von ML-Algorithmen zielt diese Dissertation darauf ab, einen Beitrag zum Verständnis und zum Schutz der Privatsphäre von Trainingsdaten zu leisten – sowohl in Bezug auf Property Inference bei bereits trainierten Modellen als auch während verteilter Trainingsprozesse.

Danksagung

Zunächst möchte ich meinem Doktorvater Hannes Federrath für die spannenden Forschungsprojekte und das Vertrauen in meine Arbeit über die letzten Jahre danken. Ebenfalls danke ich Konrad Rieck und seiner Arbeitsgruppe für das konstruktive Feedback zu dieser Dissertation.

Ich bedanke mich außerdem bei den großartigen Kolleg:innen am Arbeitsbereich Sicherheit in verteilten Systemen: Allen voran bei der besten Schreibtischnachbarschaft, die man sich wünschen kann – Johanna, für die fantastische Zeit, die wir seit Beginn unseres Studiums miteinander verbringen, Tom, für die hervorragende Zusammenarbeit und die zahlreichen Gespräche über Arbeit und darüber hinaus, und Anne, für die kurzweiligen Konversationen im Büro und den fabelhaften Espresso. Jens W., Janik und Daniel danke ich für motivierende Gespräche, produktive Brainstorming-Sessions und das gemeinsame Paperschreiben. Danke an Britta für ihre Unterstützung und auch allen weiteren Kolleg:innen für die guten Unterhaltungen und den konstruktiven Austausch: Christian, Eleftherios, Ephraim, Jens L., Kevin, Marius, Matthias, Maya, Monina, Niklas, Pascal, Stefan und Tobi. Neben Mathias Fischer möchte ich dem gesamten Team des Arbeitsbereichs Rechnernetze und insbesondere Tatjana danken, für die gemeinsame Arbeit in Projekten und die netten Gespräche zwischendurch. Für die gute Zusammenarbeit über Uni-Grenzen hinweg danke ich meinen Co-Autoren Henry, Lucas, Oliver und Julius, sowie Esfandiar, Thorsten, Moritz und Carlos.

Darüber hinaus bin ich meinen Geschwistern und meinen Eltern dankbar: Danke, dass ihr an mich glaubt und für mich da seid.

Zu guter Letzt danke ich meinem Sohn Oskar für die zusätzliche Motivation zur Fertigstellung dieser Arbeit. Und ich bedanke mich bei meiner wundervollen Frau Nele, für die Unterstützung auf allen Ebenen, das Aufmuntern zwischendurch und die Geduld mit mir und den Zahlen und Buchstaben. Es ist unfassbar schön, dass du an meiner Seite bist.

Contents

List of Figures

List of Tables

Acronyms

AI artificial intelligence 1

ANN artificial neural network 6, 8–10, 17–21, 28, 32, 37–41, 43, 47, 48, 57, 64, 65, 69–72, 74, 85–89, 91, 92, 95, 96, 136, 139, 153, 154

API application programming interface 26, 137

CNN convolutional neural network 48

DP differential privacy 4, 32

FedAvg federated averaging 24, 107

FL federated learning i, ii, 3–6, 10–14, 22–25, 30–34, 83–99, 105–113, 123, 127, 128, 135, 137, 138, 140–142

GAN generative adversarial network ix, 9, 10, 25, 32, 33, 139

HE homomorphic encryption 33, 98, 108, 111, 141

HMM hidden Markov model 28

HPO hyperparameter optimization 11, 21, 88, 93, 95, 96, 137

IG integrated gradients x, 6, 153–156

KD knowledge distillation 57, 139

KL Kullback-Leibler 29, 139, 140

LIME local interpretable model-agnostic explanations ix, x, 6, 37, 39, 53, 54, 56, 135, 153–156

LLM large language model 1, 141

LSTM long-short term memory 91

MIA membership inference attack 2–4, 9, 25, 26, 28, 32, 33, 139, 140

ML machine learning i–iii, ix, 1–3, 5, 6, 9, 11, 12, 17, 19–22, 25–28, 30, 32, 33, 37–41, 43, 55–57, 63, 65, 71, 81–87, 89, 94, 96–99, 135, 137–139, 141, 142

MNO mobile network provider 83, 84, 93, 94

NSO national statistical office 11, 82–87, 97, 98

PIA property inference attack i, ii, vii, ix, 2–10, 14, 25–29, 32, 33, 37–44, 47, 48, 50, 53, 55–57, 63–68, 70, 72–76, 135, 136, 139, 140, 142, 153–157, 160

PPML privacy-preserving machine learning 32, 63, 86, 97, 98, 141, 142

SGD stochastic gradient descent 20, 95, 126

SMPC secure multi-party computation 33, 98, 104, 108, 110–112, 127, 141

SVM support vector machine 28

t-SNE t-distributed stochastic neighbor embedding ix, 4, 6, 7, 37, 39, 54–57, 136

XAI explainable artificial intelligence 1, 4, 6, 7, 37, 39, 53, 135, 153–155

1 Introduction

Machine learning (ML) algorithms have become integral to a wide range of technological applications, playing a pivotal role in various sectors of modern society. In fact, many ML solutions are so deeply integrated into daily life that their underlying presence often goes unnoticed. Examples of interactions with ML systems which occur seamlessly in the background of other applications are

- modern search engines [Goo24],
- smartphone keyboards with autocorrect and next word prediction [HKR+18],
- voice assistants such as Apple's Siri [KB18],
- self-driving vehicles [NCC+20],
- e-mail spam filters [DBC+19], and
- recommender systems for online shopping or entertainment [PAC18].

In many domains, ML has revolutionized existing workflows or opened up new opportunities. For instance, ML has drastically improved genetic analyses, drug discovery and different areas of cancer research in the medical field [DDO+20, LHS+24, VR24].

In addition to these specialized applications, more generic large language model (LLM) chat tools such as ChatGPT have received a lot of attention recently, at the latest with ChatGPT opening to the public in November 2022 [Hu23]. Especially in the wake of the recent LLM hype, ML[1] has become a vital part of popular debates – with much of the focus on ML's capabilities, sociological implications of its use, and ethical dilemmas. However, the critical issues of data protection, privacy, and anonymity are often overlooked. Most likely, this is due to their technical complexity, lack of visibility, and the public's normalization of allowing third parties the extensive use of personal data for technological convenience. But since ML is data driven and data often contains personal information, these concerns are foundational to ML systems.

1. In public discourse, artificial intelligence (AI) is often mistakenly used as a synonym for ML. Strictly speaking, AI is an umbrella term encompassing not only ML but also other fields like rule-based systems and heuristics. Hence, I use the term ML consistently throughout this dissertation – except when referring to the established concept explainable artificial intelligence (XAI).

1.1 Problem Description

As the term “machine learning” suggests, ML algorithms are based on models which *learn* patterns and rules extracted from large amounts of training data. ML models are usually trained to fulfill a single specified task, such as predicting a future event given data from the past. Hence, a good model contains a representation of useful patterns and rules for the input data to predict a likely outcome. However, as a by-product of the learning process, other (potentially sensitive) information in the data that goes beyond these basic rules and patterns is also accumulated in the models [SRS17]. The following example is used to illustrate this problem.

Example 1. Consider an ML model for a recommender system. The model is trained and hosted by the data science department of a (fictive) company which runs an online shop. The data of purchases made by the company’s customers are used as the training data. The model’s task is to recommend an item from the shop on the input of a person’s characteristics such as their gender, age and hometown. Therefore, personal attributes of the customers are labeled with purchased store items in the training data.

Now let us assume that this model is leaked to the public – perhaps after a cyber attack on the company or due to an incautious data scientist who uploads the model to a public model store. Although the model in Example 1 is not a database itself, from which concrete customer information could be queried, it still contains a significant amount of sensitive information that can be extracted from the now published and unprotected model. Here is a brief overview of the information a potential attacker could infer from the trained model itself:

- Determine whether an individual was part of the training data, i.e., whether a person has purchased an item from the online store before. Depending on what the shop is selling, this information could be sensitive. In the literature, this is called a membership inference attack (MIA) [SSSS17].
- Infer the average traits of customers buying a certain product by launching a model inversion attack [FJR15]. While this attack primarily targets group-level attributes, it may expose individual data points in scenarios where the records associated with the same label are highly homogeneous [FJR15, SSSS17].
- Launch a reconstruction attack to reproduce actual training data samples, i.e., extracting data of individuals whose data was used during training [BCH22, CTW+21, CHN+23].
- Infer general, statistical, properties about the company’s customers via a property inference attack (PIA), such as the gender ratio [AMS+15, GWY+18]. This could be particularly interesting for competing companies.

Trained ML models “remembering too much” [SRS17] and their tendency of leaking information on many levels is an active field of research. Within the field, attacks like MIAs or model inversion attacks, being among the first published ML privacy attacks, have become quite popular. Significantly less research has been published on PIAs, which is why this dissertation addresses this gap in the literature.

But information leakage does not only occur *after* training an ML model, it may also occur *during* the training phase. This is particularly the case when training data is distributed among multiple data owners. One way to resolve the distributed scenario to train a model would be to centralize or merge the data set before training (*merge-then-train*). But when the data is valuable to the data owners, and/or it contains sensitive information, transmitting training data from one party to another is not a desirable option.

Distributed training methods are popular alternatives to the merge-then-train approach. Federated learning (FL) has become the state-of-the-art technique for this purpose [KMRR16, MMR$^+$17]. While FL allows collaborative training of an ML model without sharing the actual training data, the training algorithm requires a central coordinating party who receives model updates from the data owners (see Section 2.3 for more details on the FL training process). It has been shown that these updates can leak information about the training data from contributors [LYY20]. Additionally, data owners send their model updates to the coordinator in plain text, such that their contributions can be linked to their identities in the original FL protocol [KMRR16, MMR$^+$17].

These considerations show that sensitive information in the training data of ML models is at risk of being exposed by adversaries. They furthermore highlight the importance of developing and employing suitable protection mechanisms when using ML algorithms. This motivates the topic of this dissertation, which deals with the protection of training data during and after the training phase. The first part of this dissertation is dedicated to a thorough understanding and the exploration of defense mechanisms against PIAs in Chapters 3 and 4. The second part is focused on distributed training scenarios, practical limits of FL and the anonymity of FL participants in Chapters 5 and 6.

1.2 Research Questions and Methodology

The problem description in Section 1.1 underscores the need for more research in the areas of PIAs and distributed FL training scenarios, specifically focusing on the protection of training data during and after the training of an ML model. Therefore, I derive the following research questions for this dissertation:

RQ1 How can training data properties be reconstructed from trained ML models?

RQ2 Which strategies can help to mitigate the reconstruction of training data properties?

RQ3 What are the benefits and practical limits of FL?

RQ4 How can the anonymity of FL contributors be enhanced without sacrificing efficiency?

Different variants of the PIA have been proposed in the literature, most notably a white-box and a black-box attack scenario [GWY$^+$18, ZTO21]. In order to answer **RQ1**, both scenarios are addressed: Chapter 3 elaborates on the *white-box* scenario, in which adversaries have full access to the target model during an attack. The state-of-the-art white-box PIA [GWY$^+$18] is implemented and evaluated on three data sets. To get further insights into the effects and functionality of this attack, Chapter 3 also contains experiments with an explainable artificial intelligence (XAI) tool and the data visualization tool t-distributed stochastic neighbor embedding (t-SNE). Following up on potential shortcomings of the XAI tools in Chapter 3, the results are additionally validated in Appendix A with a broader range of tools. In contrast, Chapter 4 is mostly focused on the *black-box* PIA scenario, where the adversary can query a target model but has no access to the model's internal parameters. Additionally, PIAs are framed as a regression problem to better grasp the continuous nature of many training data properties. Chapter 4 advances the proposals from related work by introducing a novel black-box attack variant and a white-box counterpart for regression PIAs. Multiple experiments are performed to show their efficacy, including comparisons to related work. Again, three different data sets are used to validate the results.

This dissertation considers not only the PIA itself, but also possible defense approaches. As of yet, no viable defense approach has been established as a standard [RG23]. While differential privacy (DP) [Dwo06] is a key defense technique against other privacy attacks such as MIAs, it is not effective against PIAs. This is because PIAs target global properties of the training data and not nuances in a dataset caused by individual contributions that DP helps to blur [RG23, SE22]. With **RQ2**, I therefore explore novel defense strategies in both the white-box (Chapter 3) and the black-box setting (Chapter 4). The idea is to harden target models, which are at risk of being targeted by a property inference adversary, against the leakage of a specific training data property. I measure the success of a defense strategy in terms of meeting three goals:

1. It should be easy to implement, without changing the training data or model architecture;
2. defended models should leak as little information about the selected training data property as possible, i.e., the success rate of *any* PIA adversary under the respective threat model should be significantly decreased; and
3. the defense should harm the performance of target models as little as possible.

In the white-box setting, I develop a post-training adversarial learning strategy called *property unlearning* and evaluate it experimentally. Initial results show how successfully defending against a specific PIA adversary instance does not generalize to defending against other PIA instances with the same target property (violating the second goal as stated above). Hence, I investigate the strategy of consecutively employing multiple adversary instances to defend a target model through further experiments in the white-box setting. For defending against black-box attacks, I evaluate another adversarial learning strategy which hardens the model *during* the training process. This defense strategy is also evaluated experimentally regarding its efficacy and induced utility loss, exhibiting promising results. A follow-up experiment regarding its generalizability is provided in Appendix B.

While **RQ1** and **RQ2** address the protection of training data regarding different aspects of the PIA on ML models, **RQ3** and **RQ4** attend to the protection of training data on another level: protecting the data and identity of participants in distributed FL training scenarios.

RQ3 focuses on FL, its advantages and limitations. Chapter 5 is dedicated to this research question, based on the example scenario of official statistics. FL is considered as an enabler technology in this chapter, enabling the use of data and the training of ML models where company policies and data protection rules otherwise make it impossible. Three concrete application scenarios are discussed and simulated through experiments. Based on the evaluation of these three use cases, general conclusions are drawn regarding common issues and limitations when employing FL in practice.

To answer **RQ4**, existing solutions regarding the anonymity of FL contributors from the literature are grouped, analyzed and compared. Masking is a simple, yet effective way to hide individual updates from an FL aggregator: Participants can mask their individual updates with seemingly random values, the aggregator can aggregate all masked updates as a sum and subtract the sum of all masks to receive the sum of unmasked updates. While this process is simple in theory, practical issues like client dropouts during the training process demand efficient solutions to facilitate the procedure. Since pairwise- and single-masking strategies in the literature come with a high computational overhead for mask recovery if clients drop out, I introduce a novel protocol called *DealSecAgg* in Chapter 6. The protocol is evaluated and compared to related work in multiple experiments.

1.3 Contributions

Within this dissertation, various research contributions regarding the protection of training data, specifically in the context of PIAs and distributed scenarios, and the anonymity of FL training data providers are presented. This section summarizes the main contributions and explains their relation to the research questions. While Contributions **C1–C3** concern the privacy of training data *after* training an ML model, **C4** is focused on protecting training data *during* distributed training and **C5** is dedicated to the anonymity of data providers in an FL training scenario.

C1. Insights into White-Box Property Inference

Summary In this contribution, the efficacy of white-box PIAs is highlighted with experiments on multiple data sets. Then, the manifestation of property values in the weights of ML models is analyzed. Evidence shows that statistical training data properties tend to be present in different parts of the model weights, such that PIA adversaries with the same goal can deduce a property value from different parts of a target model. Furthermore, it is discovered that different training data property values can lead to substantially different model weights, such that t-SNE is able to cluster models with different property values apart.

White-Box Property Inference The state-of-the-art white-box PIA [GWY+18] by Ganju et al. was developed specifically to attack artificial neural networks (ANNs). Following their proposal, a property inference adversary trains another ANN – the so-called *meta-classifier* – to attack target models. The training data of the meta-classifier consists of the weights of hundreds or thousands of *shadow models*, i.e., models which are similar to the target model. By deliberately training these shadow models on auxiliary data sets with different properties, the adversary's meta-classifier *learns* to extract training data properties from target models. In this dissertation, the efficacy of this attack approach [GWY+18] is showcased for three data sets: MNIST, Census and UTKFace. UTKFace was not included in the original experiments by Ganju et al., for which a PIA test accuracy[2] of 99.8% is achieved in this work's experiments. For the other two data sets, test accuracies of 100% (MNIST) and 99.3% (Census) are achieved, similarly high to the results by Ganju et al.

Explaining Property Inference To the best of my knowledge, this dissertation contains the first work on analyzing the functionality of white-box PIAs. Specifically, the XAI tools local interpretable model-agnostic explanations (LIME) and integrated gradients (IG) are used to examine which parts of a target model's weights are used by the adversary's

2. Here, *test accuracy* (and later *test R^2 value* for the regression setting) refers to a value obtained by benchmarking the adversary on a test data set, opposed to measuring its performance on the training data set. In the case of PIAs, this means that not only a set of shadow models is created, but also a set of test models with training data disjunct from the shadow models is trained. While the shadow models, respectively their outputs in the black-box setting, are used for training the adversary, the test models (or their outputs) are only used for benchmarking its performance.

meta-classifier to infer a training data property. In the experiments of Chapter 3 and Appendix A, it is shown how two meta-classifiers with the same target property rely on different parts of a target model's weights. Specifically, the distributions of importance vectors – values assigned by the XAI tools to the weights of target models to measure their importance on the decision of the meta-classifier – are entirely different for two meta-classifiers attacking the *same* target model. This indicates that the influence of training data properties is not restricted to a small area of weights of a target model. Instead, such properties influence different areas of model weights, such that different PIA meta-classifier instances can infer properties through analyzing varying target model weights.

Visualizing Training Data Properties While the XAI tools show how *widespread* the influence of training data properties is on the weights of a target model, this is also the first work analyzing the *magnitude* of their influence. For this purpose, the data visualization tool t-SNE is used. t-SNE is a clustering tool: on the input of a data set, it clusters the contained data records apart, without any further knowledge about the data. The t-SNE experiments are performed with target models trained on the three data sets MNIST, Census and UTKFace. For each data set, the weights of target models with two different training data property values are used as an input for the tool. For two of the three data sets, t-SNE automatically clusters most target models apart such that they are grouped according to their training data property. Specifically, the clustering accuracy is 72.0% for the UTKFace data set and 86.8% for the MNIST data set (with a baseline for random guessing at 50%). Since t-SNE has no background information about the nature of its input data, i.e., the target model weights and their training data properties, this shows how severe the influence of training data properties can be on the weights of target models: Different property values change resulting model weights enough for an external data analysis tool, such as t-SNE, to differentiate between them.

C2. Novel Approach for Black-Box Property Inference

Summary In this contribution, a novel version of a black-box PIA is proposed: In the black-box threat model, an adversary can choose the input for a target model and observe its output, since no access to the internal weights is granted. When the attack is carried out, the adversary's input is called the *attack data set*. In contrast to related work, the attack data set the adversary uses is not part of the training data in the new proposal, hence allowing for more flexibility. Furthermore, PIAs are formulated as a regression problem rather than a classification problem in this work, as commonly seen in related research. This approach better captures realistic distributions of sensitive properties, particularly those based on ratios. Unlike classification, which requires predefined ratio classes (e.g., 40:60, 60:40), regression allows for a continuous range of values from 0:100 to 100:0, offering greater flexibility and precision. Multiple experiments are performed, generating evidence for the efficacy of this black-box attack variant. A white-box regression benchmark for the same data sets is provided, exhibiting similar performance rates.

Regression Black-Box Property Inference While black-box PIAs have been discussed in the literature before [ZTO21], this is the first attack design where the adversary can choose an arbitrary attack data set, which may be independent from the training data set. The experiments of Chapter 4 are based on the three data sets Adult, CIFAR-10 and UTKFace. In each setting, different attack data sets are used. For attacking models trained with the tabular data set Adult, an artificial data set is created. For the other two settings, data sets from a similar domain are used: parts of the CIFAR-100 data set for an attack on models trained on CIFAR-10 and another image data base for attacking models trained on the UTKFace data set. Most of related work frames PIAs as a classification problem [AMS$^+$15, GWY$^+$18, ZTO21], while recent proposals have also extended PIAs to a regression setting to better capture the nature of ratios in training data properties [SE22, ZCSZ21].

Results The proposed attack is effective in all tested settings: The black-box regression attack achieves R^2 test values of 0.72 for the Adult data set, 0.63 for UTKFace and 0.64 for CIFAR-10. While an R^2 value of 1.0 would mean that the adversary predicts each property value perfectly, 0.0 is the baseline for random guessing. Since all observed values are well above 0.5, the adversary is able to infer properties with a small margin of error.

Benchmarks For the two image data sets UTKFace and CIFAR-10, the target models of the experiments are ANNs (in contrast to Adult, where random forest classifiers are used). Hence, the UTKFace and CIFAR-10 models can be attacked by a variant of the white-box PIA by Ganju et al. [GWY$^+$18] in the regression setting for comparison. For UTKFace, the white-box regression attack achieves an even stronger R^2 test value of 0.86, while the R^2 test value for CIFAR-10 is slightly lower at 0.60. In general, the results are in a similar range for both settings. Since related work on regression PIAs unfortunately uses different data sets and/or metrics to measure the success of their attacks [SE22, ZCSZ21], only one comparison to related work could be made. One publication on classification PIAs contains a results for a “fine-grained attack” on the Adult data set [ZTO21] with five target ratios. Translating the regression results from this dissertation into a classification result similar to [ZTO21] leads to test accuracies of 40–68% (where random guessing would amount to 20%). In comparison, the accuracy results of [ZTO21] amount to similarly high 51% on average.

C3. Development and Analysis of Property Inference Defense Mechanisms

Summary This is the first published work dedicated to practical PIA defense strategies. Two different defense strategies are proposed to harden models against PIAs. They are based on the idea of *adversarial learning*, meaning that the adversary itself is used to harden a target model. For PIAs, this means that the model is retrained – during or after the initial training phase – such that it reveals less about the training data property which the adversary tries to infer. The defense success is measured in terms of efficacy (the adversary's utility loss) and the performance decrease of the target model induced by the defense. For white-box PIAs, *property unlearning* is developed and evaluated in experiments. This technique is applied *after* the initial training phase of a target model. In the experiments, property unlearning proves to be very efficient against a specific instance of a PIA adversary and does not significantly harm the target model's utility. However, it fails to generalize, i.e., protect against a whole class of PIA adversaries. In the black-box attack scenario, a similar adversarial learning approach is proposed which is applied *during* training. As with property unlearning, this approach is very effective and induces little harm regarding the target model's performance. The defense also seems to generalize well to adversaries with the same target property which were not used during adversarial learning.

Adversarial learning Adversarial learning has been originally introduced in the context of training generative adversarial networks (GANs) [GPAM$^+$14]. During GAN training, the *adversary* (usually modeled as an ANN) helps the generative model to improve the quality of its generated data by continuously revealing flaws in the generator's output. By integrating the adversary into the training process of generative models, any revealed error of implausibly generated data can be directly propagated back into the generative model, therefore improving its capabilities in each training iteration. Beyond GANs, adversarial learning has also been established as the most reliable defense strategy against adversarial examples [ACW18]. And among other applications, it has been successfully employed to defend against black-box MIAs [NSH18b], and to achieve fairness in ML models [GRLD20] as well.

White-Box Defense To defend against white-box PIAs, *property unlearning* is introduced in Chapter 3. Property unlearning is an adversarial learning strategy applied *after* training a potential target ANN. While adversarial training for GANs uses an adversary to discriminate real data from generated fake data, the PIA adversary in property unlearning is used to “unlearn” a training data property from the model weights: When applying property unlearning, the defender chooses a target property value, which may differ from the true property value of the target model’s training data. E.g., when training a model on a data set which is dominated by female records, a defender might wish to hide this fact from adversaries by setting the gender ratio target property value to 0.5, indicating that only 50% of the records are female. During the adversarial learning of property unlearning, the weights of a target model are iteratively adapted by backpropagating the “error”, i.e., the difference from the adversary’s property value prediction to the target property value, into the model. While this strategy is effective against the adversary used during adversarial training and the utility loss of target models is less than 1% on average, other adversaries with the same goal can still infer the property after it has been applied.

Black-Box Defense For defending against black-box PIAs, a similar approach is presented and evaluated. A black-box PIA adversary is used in adversarial learning, which is applied *during* the training process of the target model. This means that in every training iteration, both the usual training loss of a model is calculated to optimize its weights for the model’s task, but also the adversarial loss, i.e., the distance of the adversary’s prediction to the target property value, is computed. Both loss values are combined to calculate the model’s gradient via backpropagation, with a parameter $\lambda \in [0,1]$ controlling the influence of the adversarial loss. $\lambda = 0$ would amount to ordinary model training with no adversarial influence, while $\lambda = 1$ means that the model is only trained to make the adversary predict the target property value, ignoring the model’s original task. In the experiments of Chapter 4, a good tradeoff between model utility and defense is found at $\lambda = 0.15$: While the model utility stays in the same range across both tested data sets (UTFace and CIFAR-10), the adversary’s performance decreases from an R^2 of 0.6 to 0.07. Additional experiments in Appendix B show how this defense strategy also generalizes well to other adversaries with the same goal, which were not included in the adversarial learning process. For the other adversarial instances (trained on different shadow models than the adversary used in adversarial learning), the R^2 performance drops from 0.6 to 0.3 on average.

C4. Analysis of Benefits and Practical Limits of Federated Learning

Summary In this contribution, an analysis of the potential of FL is provided, framed in a case study regarding official statistics. Simulations for three different use cases in the areas health, sustainability and mobility are evaluated. The first two use cases focus on the performance comparison of an ML model trained on centralized data and a variant of the same model trained via FL in a distributed manner. The third use case is situated in a more realistic environment, where the training data remains private, meaning that there was no training data available during model development. Therefore, the evaluation provides insights on the performance of FL in different use cases, as well as it discovers limitations and common issues when applying FL in practice.

Official Statistics In the domain of official statistics, national statistical offices (NSOs) aim to produce high-quality statistics that accurately reflect reality. To keep up with a rapidly changing world, NSOs are modernizing by adopting new data sources, methodologies, and technologies. Similar to many other domains, NSOs are increasingly adopting ML technologies. But as of now, many potential data sources, such as private or official entity data, remain untapped due to slow legal adaptation and privacy concerns. In this context, FL is a particularly interesting technology, since this decentralized technique enables a privacy friendly data analysis, opening up new use cases while maintaining a basic data privacy. The study presented in Chapter 5 aims to investigate the potential of FL in the context of official statistics as an enabling technology.

Use Cases The case study explores three use cases to evaluate the effectiveness of FL compared to centralized models. In predicting medical insurance charges, the best centralized model achieved an R^2 of 0.85, while the FL model reached 0.78, demonstrating a small but acceptable performance gap in exchange for improved data privacy. For fine dust pollution classification, the centralized approach achieved 72% classification accuracy, whereas the FL model performed slightly lower at 68%, still proving its viability for environmental monitoring. The third use case, predicting users' daily movement radius from mobile network data, showed weaker overall model performance, with the best centralized approach achieving an R^2 of 0.16 and the FL model reaching 0.11. Despite its challenges, FL consistently delivered results close to centralized models, making it a promising approach for privacy-sensitive applications in official statistics.

Key Insights The study highlights that FL can achieve performance close to centralized models while preserving data privacy, making it a valuable approach for official statistics. Although a slight, but expected, performance gap is observed due to the decentralized nature of FL, it remains an effective solution, especially in domains where data sharing is restricted. The results also emphasize the importance of hyperparameter optimization, which remains a challenge in FL due to limited automated optimization tools. Additionally, the study underscores the practical difficulties of implementing FL in real-world scenarios, such as technical barriers, communication costs, and the need for efficient infrastructure. Despite these challenges, FL enables statistical offices to access and analyze sensitive data without violating privacy regulations, potentially accelerating the adoption of new data sources in official statistics.

C5. Novel Protocol for the Anonymity of Data Providers in Federated Learning

Summary While FL enables multiple data owners to train an ML model collaboratively without exchanging training data, a central party is responsible for orchestrating the training process. This central party is called the aggregator, since their main tasks are collecting and aggregating model updates from the participants and redistributing the updated global model in each training round. The aggregator has no access to training data. However, private information can also be reconstructed from individual model updates. Since the aggregator receives these updates directly from the FL participants, i.e., the data holders, mapping received information to an individual contributor is trivial. Although there are solutions in related work which create an anonymity group around all FL contributors, they all lack communication or computational efficiency, especially when clients drop out before completing a training round. Hence, a novel masking protocol called *DealSecAgg* is proposed. By introducing a dealer party, cryptographic operations are outsourced from both the aggregator and clients. The protocol only needs two communication rounds (strictly less than related work) and its runtime is not increased if clients drop out during execution. The dealer party never has access to any training data or model updates, thus little trust is needed – apart from the requirement that the dealer does not collude with the aggregator. A variant of DealSecAgg with multiple dealers is also presented, in which all but one dealer may collude with the aggregator to maintain the protocol's integrity. Multiple experiments are performed to verify the reduced runtimes, improved dropout resilience and enhanced communication efficiency compared to related work. A security proof is also provided.

Federated Learning Masking Strategies In FL, masking strategies are a popular choice to ensure that individual client updates remain private while still allowing for cryptographically secure model aggregation. By adding random noise, or "masks", to updates before transmission, these techniques prevent attackers from linking sensitive data to specific users. Pairwise-masking, as used in SecAgg [BIK$^+$17] and SecAgg+ [BBG$^+$20], generates shared masks between client pairs, ensuring that when updates are summed, the masks cancel out. While effective, this approach is computationally demanding, leading to optimized variants like FastSecAgg [KRKR20], which reduces complexity using a secret-sharing scheme. Single-masking, introduced in LightSecAgg [SNY$^+$22], simplifies the process by allowing unmasking in a single step but introduces additional encoding overhead. To address these overheads, DealSecAgg is introduced to improve efficiency by outsourcing mask aggregation to a separate dealer, reducing computational burden while maintaining privacy guarantees.

DealSecAgg Protocol The DealSecAgg protocol, as introduced in Chapter 6, enhances secure aggregation in FL by introducing a dealer-assisted masking strategy that reduces computational overhead while preserving privacy. Instead of requiring clients to establish pairwise masks, each client generates a single random mask and sends only the mask key to a trusted dealer. The dealer then reconstructs and aggregates the masks without ever accessing the model updates themselves. Clients transmit their masked updates to the aggregator, which receives the aggregated mask from the dealer and subtracts it to recover the sum of all client updates. This process requires only two communication rounds, making DealSecAgg significantly more efficient than previous protocols. A multi-dealer variant further strengthens security by ensuring that privacy remains intact as long as at least one dealer does not collude with the aggregator. By offloading mask management to an independent entity, DealSecAgg minimizes the computational burden on clients and reduces communication overhead while maintaining strong privacy guarantees.

Experimental Results The experimental results demonstrate that DealSecAgg significantly improves efficiency compared to existing secure aggregation protocols. When tested on FL with the CIFAR-10 dataset, the protocol achieved the same accuracy as standard federated averaging, confirming that privacy-preserving masking does not degrade model utility. In terms of computational efficiency, DealSecAgg reduced runtimes by up to 87.8% compared to LightSecAgg [SNY+22] and even 97.6% compared to SecAgg+ [BBG+20], particularly benefiting resource-constrained client devices. The protocol also scaled well with increasing numbers of clients, dropouts, and dealers, maintaining low computational costs even as complexity grew. Communication overhead remained comparable to SecAgg [BIK+17] and SecAgg+, with DealSecAgg requiring only two communication rounds per training step, significantly reducing delays in high-latency networks. The results confirm that DealSecAgg is a scalable solution for secure FL offering strong privacy protection with minimal computational and communication costs.

1.4 Structure of this Dissertation

As summarized in Table 1.1, this dissertation is structured as follows: Chapter 2 provides general background information for the following chapters. The main part is divided into two parts: The first part focuses on PIAs, covering white-box attacks in Chapter 3 and black-box attacks in Chapter 4. The focus of the second part is FL, with Chapter 5 exploring strengths and practical limits of FL and Chapter 6 proposing a masking scheme for the anonymity of FL participants. The dissertation is concluded by a summary and an outlook in Chapter 7. In Appendices A and B, the results of additional experiments are presented, focusing on white-box PIA explainability and the black-box PIA defense.

Area	Chapter	RQs	Contributions	Publication
	Chapter 2: *Background*			
Property Inference Attacks (PIAs)	Chapter 3: *White-box PIA, Insights and Defense*	**RQ1**, **RQ2**	**C1**, **C3**	[SWDF23]
	Chapter 4: *Novel Black-Box PIA, White-Box Comparison and Defense*	**RQ1**, **RQ2**	**C2**, **C3**	[SLRF24]
Federated Learning (FL)	Chapter 5: *Benefits and Practical Limits of FL*	**RQ3**	**C4**	[SHWF23]
	Chapter 6: *Efficient Protocol for the Anonymity of FL Participants*	**RQ4**	**C5**	[SHSD24]
	Chapter 7: *Conclusion*			

Table 1.1: Overview of chapters and their connection to research questions, contributions and publications.

1.5 List of Publications

The following is a list of the publications included in this cumulative dissertation.

[SWDF23] Joshua Stock, Jens Wettlaufer, Daniel Demmler, and Hannes Federrath. (2023). Lessons Learned: Defending Against Property Inference Attacks. In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT)*. ISBN: 978-989-758-666-8, ISSN: 2184-7711, SciTePress, pages 312–323. DOI: 10.5220/0012049200003555

[SLRF24] Joshua Stock, Lucas Lange, Erhard Rahm, and Hannes Federrath. (2024). Property Inference as a Regression Problem: Attacks and Defense. In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT)*, ISBN: 978-989-758-709-2, ISSN: 2184-7711, SciTePress, pages 876–885. DOI: 10.5220/0012863800003767

[SHWF23] Joshua Stock, Oliver Hauke, Julius Weißmann, and Hannes Federrath (2023). The Applicability of Federated Learning to Official Statistics. In *Intelligent Data Engineering and Automated Learning (IDEAL)*. Lecture Notes in Computer Science, vol 14404. Springer, Cham, ISBN: 978-3-031-48232-8, pages 70–81. DOI: 10.1007/978-3-031-48232-8_8

[SHSD24] Joshua Stock, Henry Heitmann, Janik Noel Schug, and Daniel Demmler (2024). DealSecAgg: Efficient Dealer-Assisted Secure Aggregation for Federated Learning. In *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES)*. Association for Computing Machinery, New York, NY, USA, ISBN: 979-8-4007-1718-5, Article 183, pages 1–11. DOI: 10.1145/3664476.3670873

I also contributed to the following articles and research papers that are not included in this dissertation:

[SPB$^+$22a] Joshua Stock, Tom Petersen, Christian-Alexander Behrendt, Hannes Federrath, and Thea Kreutzburg. Privatsphärefreundliches maschinelles Lernen: Teil 1: Grundlagen und Verfahren. *Informatik Spektrum*, 45(2):70–79, 2022. DOI: 10.1007/s00287-022-01438-3

[SPB$^+$22b] Joshua Stock, Tom Petersen, Christian-Alexander Behrendt, Hannes Federrath, and Thea Kreutzburg. Privatsphärefreundliches maschinelles Lernen: Teil 2: Privatsphäreangriffe und privacy-preserving machine learning. *Informatik Spektrum*, 45(3):137–145, 2022. DOI: 10.1007/s00287-022-01440-9

Continued on next page

[KPS+24] Moritz Kirschte, Thorsten Peinemann, Joshua Stock, Carlos Cotrini, and Esfandiar Mohammadi. S-GBT: Frugal differentially private gradient boosting decision trees. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2024. DOI: 10.1145/3658644.3690301

[MBG+23] Johanna Ansohn McDougall, Alessandro Brighente, Willi Großmann, Ben Ansohn McDougall, Joshua Stock, and Hannes Federrath. LoVe is in the Air - Location Verification of ADS-B Signals Using Distributed Public Sensors. In *IEEE International Conference on Communications (ICC)*, pages 6040–6045. IEEE, 2023. DOI: 10.1109/ICC45041.2023.10278848

[CSR+20] Rosario Cammarota, Matthias Schunter, Anand Rajan, Fabian Boemer, Ágnes Kiss, Amos Treiber, Christian Winert, Thomas Schneider, Emmanuel Stapf, Ahmad-Reza Sadeghi, Daniel Demmler, Joshua Stock, Huili Chen, Siam Umar Hussain, Sadegh Riazi, Farinaz Koushanfar, Saransh Gupta, Tajan Simunic Rosing, Kamalika Chaudhuri, Hamid Nejatollahi, Nikil Dutt, Mohsen Imani, Kim Laine, Anuj Dubey, Aydin Aysu, Fateme Sadat Hosseini, Chengmo Yang, Eric Wallace, Pamela Norton. Trustworthy AI Inference Systems: An Industry Research View. *arXiv preprint*, 2020. DOI: 10.48550/arXiv.2008.04449

2 Background

The umbrella term machine learning (ML) describes a class of algorithms to build predictive models from training data. Building on the definition in [Zho21], a data set generally consists of multiple samples. For supervised learning[1], each sample comes with a label. A data set DS_t thus contains pairs (x_i, y_i), where $y_i \in Y$ is the label of the ith sample $x_i \in X$ and Y is the set of all labels (the *output space*). A model establishes a mapping $f : X \mapsto Y$ from the *input space* X to the output space Y by learning from a training data set DS_t. The output space depends on the type of prediction problem at hand: For *classification* problems, the output space is discrete, while the output space is continuous for *regression* problems.

Let us revisit Example 1 of Section 1.1 and apply the notation above. Here, the model for the recommender system maps customer data X to purchased items Y. Hence, the output space is discrete and the model solves a classification problem. This dissertation primarily discusses classification problems, with the exception of Chapter 4.

The key to the success of ML models is their automated optimization – the *training phase*. During this phase, a model is *fit* to training data by algorithms. In other words, the model "learns" to make correct predictions based on the examples contained in the training data. Ideally, this ML model can be used afterwards to classify (respectively regress on) previously unseen data, a property which is called *generalization* in the literature [Zho21]. The lifespan of an ML model, including its training, is discussed in Section 2.2.

The following sections provide the necessary background information relevant for Chapters 3 to 6, the research papers of this cumulative dissertation. Although brief introductions to some of the topics are also provided within these publications, this chapter is meant to provide a comprehensive and more detailed overview of the covered topics. First, a popular type of ML model is introduced.

2.1 Artificial Neural Networks

While there are many different types of ML models, this dissertation is mainly concerned with ANNs. On one hand, ANNs are generic in their design, which enables them to solve a variety of problems. On the other hand, ANNs are extremely powerful, which has

1. The scope of this dissertation is limited to supervised learning. Although the research in this dissertation could be applied to other fields such as unsupervised learning (ML with unlabeled data, e.g., for data clustering), but much of related work is limited to supervised learning which motivates the scope here.

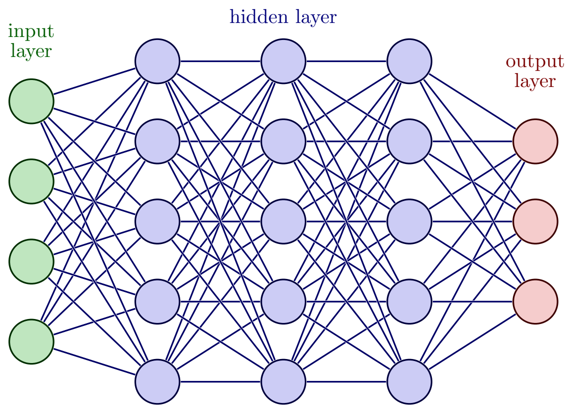


Figure 2.1: ANN schematic diagram. By Izaak Neutelings, licensed under CC BY-SA 4.0.

been proven in many competitions[2]. The following introduction to ANNs is based on [Zho21].

As their name suggests, ANNs are based on neural networks, i.e., biological nervous systems like the human brain. At the core of such networks are *neurons*, which are interconnected. In a biological network, a neuron sends neurotransmitters to its neighbors whenever it is "excited", changing their electric potentials. In turn, the excitement (or *activation*) of a neuron occurs when the electric potential exceeds a certain threshold. The idea to apply these concepts in a mathematical context dates back to the 1940s [MP43]. But while natural evolution has found a way to logically adjust the neuron activation thresholds in biological networks via *learning*, training algorithms for ANNs were computationally too expensive for generally available hardware of the 20th century. It took until around 2010, when the significant advantages of cheap computing power and big data started to make the use of ANNs more popular [Zho21].

In most modern ANNs, neurons are structured by layers, with activations flowing from an input layer through an arbitrary number of "hidden" layers to an output layer. In a fully connected neural network, a special case of such an ANN, the neurons of one layer are all interconnected with the neurons of its neighboring layers, but there are no connections between neurons within a layer. This is shown in Figure 2.1.

Artificial neurons are typically equipped with non-linear activation functions such as *sigmoid* or *ReLu* (rectified linear unit) [Zho21]. Each neuron assigns a variable *weight* value w to each incoming edge from other neurons. As shown in Figure 2.2, the output of a neuron is calculated as follows: The activation function σ is applied to the sum of all weighted inputs from neighboring neurons and a variable *bias* value b. The weights

2. As a popular example, variants of ANNs have won the *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* many times: https://www.image-net.org/challenges/LSVRC/ (accessed on February 3, 2025)

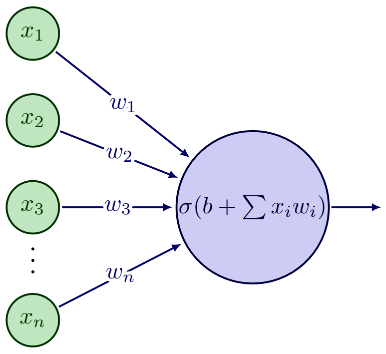


Figure 2.2: The activation of a neuron in an ANN is computed by summing up its n weighted inputs $x_i w_i$, adding its bias value b and applying the activation function σ. Adapted from Izaak Neutelings, licensed under CC BY-SA 4.0.

w_i and bias values b of all neurons make up the trainable *parameters* of an ANN. More details on how the parameters are trained are provided below.

2.2 The Machine Learning Pipeline

Training a model is at the core of the ML data flow. To put the training phase into perspective, the entire lifespan of an ML model is described in this section and illustrated in Figure 2.3. The data flow is called a *pipeline* [HN20] and is made up of the following steps[3]:

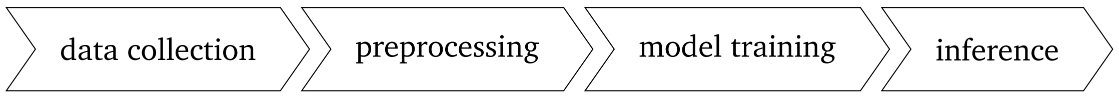


Figure 2.3: The ML pipeline.

Data Collection and Labeling The first step of the ML pipeline is the collection of data. Often data is collected automatically, e.g., orders of an online shop are entered by its customers and collected in a data base. The result of this step is a data set DS_u.

Depending on the data set, the labels $y_i \in Y$ might be an integral part of the data, such as in Example 1. In other scenarios however, labeling is a manual or semi-automated and often expensive process [HZ24], especially if domain experts need to be payed to perform the labeling. This is often the case for medical or image data.

3. While the term ML pipeline usually refers to reproducible, automated workflows in the literature [HN20], it is used in an abstract way here, covering the manual execution of the steps as well. The number and scope of the steps may vary depending on the source in the literature.

Data Preprocessing and Feature Engineering DS_u may still lack a certain degree of integrity, which can get in the way of training algorithms. To ensure usability of the data, it is often necessary to apply preprocessing steps. Depending on the data set condition, these steps might include:

- cleaning (removing outliers or faulty data records),
- handling missing data (interpolation or removal of incomplete data records),
- standardization (alignment of image data, scaling numeric data, converting textual attributes to class labels or numbers), and
- feature extraction: In more complicated settings, e.g., when text or audio is being processed, features need to be extracted from the data prior to training. For text classification, this could include introducing word embeddings, as a way of discretizing natural language into vectors of word embeddings to make the natural language machine readable for the ML model [WWC+19].
- Feature selection, the choice of appropriate attributes and features from the preprocessed data, can be considered as the last step in preprocessing.

Preprocessing can be framed as a transformation from a data set DS_u to a training data set $DS_t \subseteq X \times Y$.

Model Training After preprocessing, the model can be trained with the training data set DS_t, consisting of pairs (x_i, y_i). The model is trained to output the correct label $y_i \in Y$ on input of a data sample $x_i \in X$. A *loss function* quantifies the error of a model, i.e., how much the model diverges from its optimal behavior (correctly predicting the respective label of every sample in the training data). An example loss function for regression problems is the *mean squared error* (MSE) which calculates the error between the true labels (as in the training data) and the output of the model. The better a regression model is trained to solve the task at hand, the lower is its resulting MSE. In general, the objective is to minimize a model's loss function during training. This can also be called the *empirical risk minimization* approach, where the goal is to find model parameters that minimize the risk or an objective function (such as the loss function) of the model [Vap91].

Modern ANN training algorithms use *backpropagation*, which typically relies on stochastic gradient descent (SGD) or similar techniques to optimize the model's parameters [Zho21]. In simple terms, SGD examines the output of the loss function and calculates the direction in which the model's parameters should be adjusted to minimize the loss. This process involves *propagating back* the error calculated by the loss function into the model. Since the direction for adjusting parameters provided by SGD (indicated by the term *descent* in SGD) is only applicable to the model's current state, these adjustments must be made cautiously. If the parameter values are updated too drastically, it can cause the model to skip over a local minimum of the loss function, potentially leading to an increase in the loss. The amount of induced change is regulated by the *learning rate* $\eta \in (0, 1)$. In order to slowly approach a (local) optimum of the parameter values, training processes are always iterative, where a single round of training is called *epoch*. A lower learning rate generally means that the training process will need more epochs to converge to a local optimum, but increases the chance of actually finding a good optimum. While the details of the backpropagation algorithm are out of scope of this dissertation, the interested reader can be pointed to [Zho21, Chapter 5.3].

Before starting the training process, an ML model m must be constructed. For neural networks, this means that their architecture must be designed: This includes the number of layers, the number of neurons per layer and the type of activation function. Additional hyperparameters concern the training process and must also be decided on [Zho21]:

- the learning rate η (controlling how much model parameters are adapted in each training step; might also be adjusted during the training process),
- the batch size (training data samples are typically not processed individually but in *batches* for performance reasons),
- the loss function (such as *categorical crossentropy* for classification or *mean squared error* for regression problems),
- and the number of training epochs.
- Optionally, early stopping mechanisms can be provided, e.g., to stop training when a certain performance threshold has been reached.

Choosing these hyperparameters is not trivial. In order to find optimal hyperparameters for a given problem, there are automated tools in the area of hyperparameter optimization (HPO), which can perform a grid search over a predefined parameter space, for example [FH19]. When the model is designed and hyperparameters for its training are set, each neuron in an ANN must be initialized with pre-defined or random weights. Then, the training itself, i.e., the iterative tuning of the neurons' weights and biases, can begin.

During and after training epochs, the model's performance is typically evaluated using *test data*, which is a separate dataset not involved in the training process. Carefully selected test data can help to discover *overfitting*, i.e., when a model is tailored well to specific patterns in the training data, but does not generalize to other data. A clear sign of overfitting is when the loss of a model is low for training data, but error rates on test data are much higher [Zho21].

Inference After a model has been trained, it can be used for its designated prediction task. Depending on the context, it might be used internally in the model owner's organization, or it could be embedded into software, deployed on a server, published, archived, etc. For some use cases, it could be important to update the trained model regularly, e.g., if the data used as training data is changing frequently. Since predictions can be *inferred* with the model, this step is often referred to as the inference phase.

2.3 Distributed Training Data

In many cases, data sets are stored on multiple premises, and not in one single place [KMRR16]. Combining this data to train an ML model on the entirety of the data can often unlock large potentials, as the examples in Section 2.3.2 below will show. In order to train a model on this decentralized data, the most straightforward approach is to combine the data into one data set and let one party train the model on this aggregated data set (*merge-then-train*). The disadvantage of merge-then-train is that a single party gains access to the whole data set. Since data potentially includes sensitive and/or personal information, this approach is often unfavorable. As an alternative, decentralized learning techniques can be applied, such that no data records need to be transferred to a central party.

Some decentralized model training approaches are in the area of *fully decentralized learning* or *peer-to-peer learning*, such as gossip learning [OHJ13, HDJ19] or swarm learning [WHSS+21]. These techniques rely on data owners within a network without orchestrating central parties, where trained models are exchanged with neighboring nodes. However, the focus in this dissertation is on federated learning, a much more popular strategy which involves a central coordinating party.

2.3.1 Federated Learning

Federated learning (FL) enables multiple data owners to train an ML model in a decentralized way [KMRR16, MMR+17]. While different versions of FL exist, the focus of this work is on horizontal FL, in which the local data sets of all data owners share the same (or similar) features but contain different samples. This is in contrast to vertical FL, where a data set is distributed in its features but concerns the same samples across data holders [LYY20].

A schematic overview of FL is illustrated in Figure 2.4. The data owners, also referred to as *clients* or *participants*, initially receive a model from the central party, known as the *aggregator*. Upon receiving the initial model, clients begin training for a pre-determined

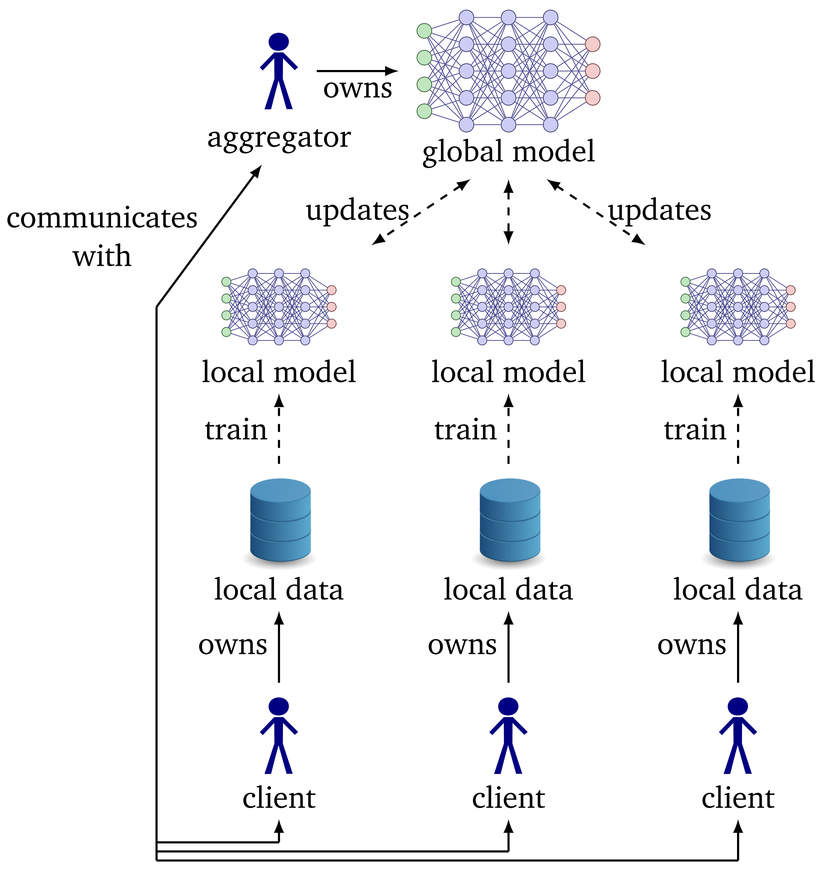


Figure 2.4: Schematic diagram of FL. The dashed lines represent data flows, while full lines represent possible actions. This graphic is inspired by [RG23] and includes content by Izaak Neutelings, licensed under CC BY-SA 4.0.

number of epochs on their local data before returning their updated models. Once the aggregator has received all client updates, it combines them to compute the next iteration of the global model. Typically, the average update for each weight (calculated across all client updates) is used, which is known as federated averaging (FedAvg) [MMR+17]. The updated global model is then redistributed to the clients to start the next decentralized training round. Depending on the context, the set of participating clients may vary in each round. E.g., if the pool of potential clients is large, a different subset of participants may be selected in each round to minimize computational effort while ensuring diversity in the training contributions.

A formal definition based on [MMR+17] is presented in Section 6.2: Generally, n FL clients have the goal of training a global model g. In each training round τ, a set of clients C_τ is chosen from a pool of data holders such that $|C_\tau| \leq n$. After their local training, the aggregator combines the clients' model updates u to calculate the new global model for the next round $\tau + 1$ via FedAvg:

$$g_{\tau+1} = \sum_{c_i \in C_\tau} \frac{u_i}{|C_\tau|}, \text{ with model update } u_i \text{ of client } c_i.$$

2.3.2 Applications of Distributed Training

A popular FL application from the real world is Google's mobile keyboard prediction model *Gboard* [HKR+18]. Installed on millions of Android devices worldwide, keyboard inputs are stored locally. Updates to the Gboard models are trained overnight with 100 participating devices in each training round [HKR+18]. Participants may be chosen if they are connected to a battery charger and have an active WiFi connection. The global model is redistributed to the devices after training and customized according to individual user inputs. FL is easily scalable beyond 100 clients, as demonstrated in the original proposal paper [KMRR16]. Here, experiments with up to 10000 clients are discussed for the use case of predicting comments in a social network.

In reference to Example 1, let us transfer the fictive scenario to a distributed setting:

Example 2. Consider the model owning company partnering up with other companies who run online shops selling similar products. The idea is to combine the data of all partnering companies, in order to train a model on a more comprehensive data set. Since sharing customer data with competitors is not an option, FL can be applied – using an independent trade association as their coordinating party. Now, they can cooperatively train a model on their combined data sets, without exchanging actual data records. The resulting model can then be used by all participants after the training process has been completed.

The technical requirements and challenges of such a collaborative training scenario are covered in Chapter 5.

2.4 Privacy Risks in Machine Learning

Both centralized and distributed ML can come with risks for the privacy of data owners and data subjects, as numerous publications have shown [LYY20, RG23]. Since privacy and security in the context of ML have become a large field of research, the focus of this dissertation is on selected scenarios: The possibility of extracting information from a trained model, specifically PIAs, and training a model in a distributed way via FL. For these scenarios, threat models and relevant attacks are summarized below.

2.4.1 Information Leakage of Machine Learning Models

When trained models are made accessible to the public, adversaries may exploit them to extract information beyond their intended use, such as solving classification or regression tasks. Research has shown that models often disclose more information about their training data than the model owners might expect [RG23, LWH+22]. While overfitting models are more likely to leak information, it has been demonstrated that overfitting is not a necessary condition for these leaks [YGFJ18].

One of the first relevant works in this line of research was by Shokri et al. They have introduced the membership inference attack (MIA) [SSSS17], i.e., an attack to decide whether a specific data sample was part of a target model's training data. While Shokri et al. leverage a target model's black-box access in the centralized training setting, multiple publications have extended this approach, e.g., to white-box and distributed settings [NSH18a] or to other model types like GANs [CYZF20, HMDDC17, HHB19].

There are several other well-studied forms of information leakage in trained ML models, including reconstruction attacks [BCH22, CTW+21, CHN+23], model inversion [FJR15], or model stealing in black-box scenarios [JSMA19]. Aiming to address a gap in this area of research, this dissertation focuses on PIAs.

2.4.2 Threat Models for Property Inference

For the centralized training data setting, the data owner (as an individual or as an organization) is assumed to have access to all training data. The ML model is assumed to be created, trained and used on premise, such that no data is transferred to other parties and the data owner is also the model owner. Hence, more complex attack surfaces such as external services for model training or insider attacks within the data owner's organization are out of scope. Poisoning attacks, where the adversary may control parts of the training data, are also out of scope.

In this dissertation, the threat model for centralized learning is focused on model owners with a published ML model. Potential adversaries have access to the trained and published model as well as to general information about the training data or to a small excerpt from it[4]. In its simplest form, publishing a trained model means uploading a model file to the internet, i.e., releasing all trained weights and biases of a model. When adversaries can read these weights and biases, e.g., by downloading them from a website like a legitimate model consumer, they have *white-box* access to the model as illustrated in Figure 2.5.

More careful approaches to publishing an ML model usually offer an application programming interface (API) to consumers outside of the model owner's organization. Through this API, consumers can then query the model with input data to receive its predictions. In such a *black-box* scenario, the model owner stays in control of the model and can limit its access, as depicted in Figure 2.6. For instance, model access could only be granted to registered users, queries may be logged and rate limiting can be applied, such that only a certain number of queries can be executed per time interval and user. Both white-box and black-box threat models are common and realistic [SLCE23], thus both are considered in this dissertation.

2.4.3 Property Inference Attacks

The property inference attack (PIA) was introduced by Ateniese et al. in 2015 [AMS⁺15]. In contrast to other types of information leakage, like MIAs, a PIA is aimed at *global* statistical properties of a training data set DS_t. These global properties can take many different forms, as the following examples show:

- sentiments in emails as the training data of a spam classifier, or the volume of a company's transactions from a fraud detection system [MGC22],
- the most dominant medical specialty in the training data leaked from a rating-prediction system for health care text reviews [ZTO21], or

4. While the adversary might not have access to parts of the training data in realistic scenarios, this assumption is made in various other publications [NSH18b, SM21, TMS$^+$21]. However, information about the training data can also be reconstructed through separate attacks [SSSS17], even without having access to any training data, which has been shown to be just as effective [LWH$^+$22].

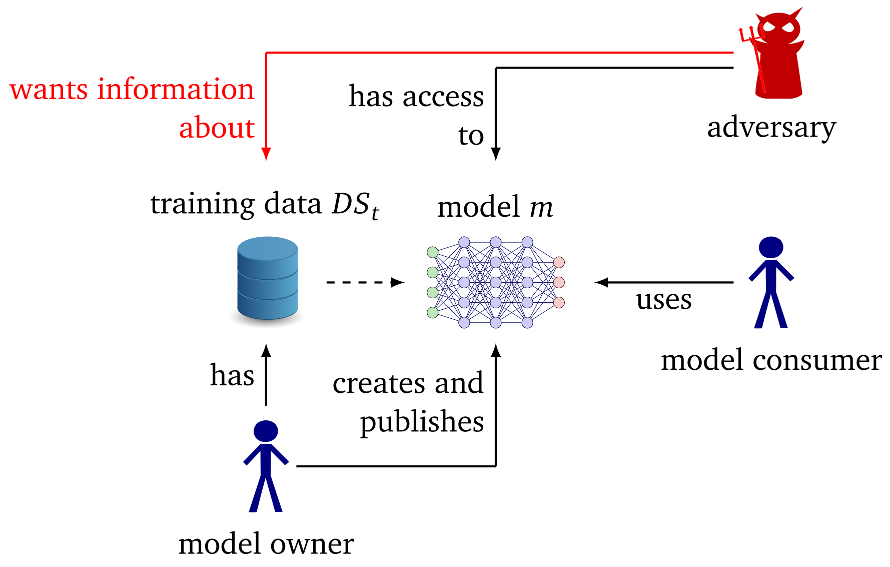


Figure 2.5: White-box threat model for PIAs against ML systems. The dashed line represents a data flow, while full lines represent possible actions. This graphic is inspired by [RG23] and includes content by Izaak Neutelings, licensed under CC BY-SA 4.0.

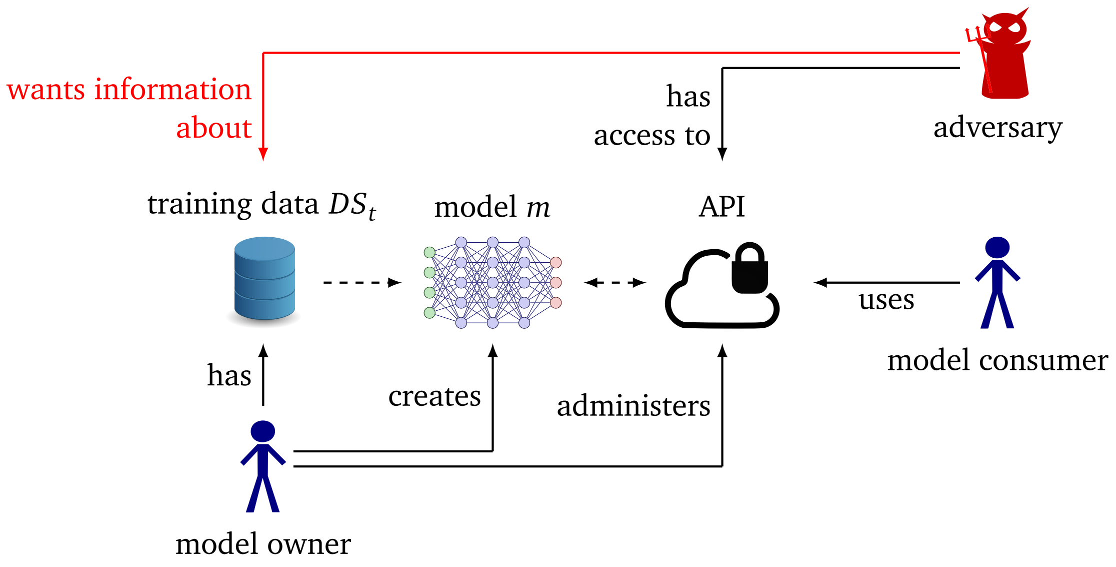


Figure 2.6: Black-box threat model for PIAs against ML systems. Dashed lines represent data flows, while full lines represent possible actions. This graphic is inspired by [RG23] and includes content by Izaak Neutelings, licensed under CC BY-SA 4.0.

- the presence of a dialect in the training data of a speech recognition system, respectively nuanced traffic information from the training data of a coarse network traffic classifier [AMS$^+$15].

- In [GWY$^+$18], the authors identify whether a system has been patched against specific security breaches given a cryptocurrency mining detector trained on hardware performance counters. They also infer the gender ratio from an income predictor, the level of noise for images in the training data of a digit classifier, and various demographic properties from a smile detecting model and a gender classifier.

PIAs can also be used as a building block for sophisticated MIAs [ZCSZ21]. And beyond the standard attack scenario in which an adversary maliciously gathers information, a PIA can also serve as a tool for fairness audits – since global properties of a training data set can indicate bias of a target model[5].

The original PIA proposal is aimed specifically at the leakage of hidden Markov models (HMMs) and support vector machines (SVMs) in a white-box threat model [AMS$^+$15]. Ganju et al. adapted the approach to attack ANNs. Like Shokri et al.'s MIA [SSSS17], both PIA proposals use the technique of shadow model training [GWY$^+$18], involving the training of multiple *shadow models* which each mimic the target model. The weights of the shadow models are then used as a training data set for the adversary's property inferrer – which is implemented as an ANN itself. A detailed description of the state-of-the-art white-box attack by Ganju et al. [GWY$^+$18] is described in Section 3.2.4. Different flavors of the black-box attack are presented in Section 4.2.

Our formal definition of a PIA[6] is based on [SE22]. Both the model owner and the adversary $\mathscr{A}$ have access to a public distribution of data $D = (X, Y)$ with data X and corresponding labels Y. Both parties also have access to two distribution transforming functions G_0 and G_1. Property inference is now defined as a game in which the model owner trains a model by using training data from either of the distributions $G_0(D)$ or $G_1(D)$, i.e., $DS_t \leftarrow G_0(D)$ or $DS_t \leftarrow G_1(D)$. After training model m with DS_t, the trained model is released to the adversary. The task of the adversary is to infer whether m was trained with $G_0(D)$ or $G_1(D)$, i.e., to infer the value $b = \{0, 1\}$ of $G_b(D)$.

5. There is a large body of research around the topic of fairness in ML which is out of scope for this dissertation [MMS$^+$21]. A popular example for bias in a real-world ML system is demonstrated in [ALML16].
6. This definition covers the classic way of interpreting property inference as a classification problem, i.e., the adversary infers a property value from a limited set of values. While most of the PIA literature follows this approach (including [AMS$^+$15] and [GWY$^+$18]), Chapter 4 is focused on PIAs as a regression problem. When formulating property inference as a regression problem, the adversary tries to infer a property value from a continuous range of values, e.g., from the range of 0 to 1.

After receiving m, the adversary uses algorithm H to make its prediction $\hat{b}$ of $G_{\hat{b}}(D)$. Therefore, we can define the adversary's advantage by using H in the following way:

$$\mathrm{Adv}_H = |\Pr[\hat{b}|b=1] - \Pr[\hat{b}|b=0]|.$$

Example 3. Revisiting the domain of Example 1 once more, the distribution $D = X, Y$ would be defined over customer data X and product labels Y. A PIA could be targeted at the gender distribution of customers, with $G_0(D)$ defined as the transformation over D such that 80% of the customer data is male and $G_1(D)$ for an equal split of 50% males and 50% females. The adversary thus wants to infer whether the company's customers are predominantly male or of a balanced nature.

By basing the definition on publicly known functions which work on the same underlying distribution D, a wide range of possible attack goals and scenarios (even beyond PIAs) can be captured [SE22].

A new approach to black-box PIAs called *Kullback-Leibler (KL) divergence* has been recently introduced by Suri et al. [SLCE23]. For this KL divergence attack, the adversary first samples data from $G_0(D)$ and $G_1(D)$. These two auxiliary data sets are then used to train multiple shadow models M_0 and M_1, just like in the other attacks mentioned above. The adversary uses an attack data set DS_{att} to query both the shadow models and the target model m. Now for all pairs of shadow models $(m_0, m_1) \in M_0 \times M_1$, the estimated KL divergence from m to m_1 is subtracted from the divergence of m to m_0. The divergences are calculated using the outputs of the models on the input of DS_{att}. Then, the adversary calculates the sum of all the differences calculated above

$$\sum_{m_0^i \in M_0} \sum_{j_1^j \in M_1} KL[m(x), m_0^i(x)] - KL[m(x), m_1^j(x)] \tag{2.1}$$

with $x = DS_{\text{att}}$ and the KL divergence KL. If the result[7] of Equation 2.1 is positive, the sum of all divergence differences between the outputs of $m_1 \in M_1$ and m is smaller than the sum of divergences between $m_0 \in M_0$ and m – thus the adversary predicts $\hat{b} = 1$, and vice versa. The main advantage of this attack is that significantly less shadow models need to be trained. Suri et al. claim that the attack works with as little as 5 shadow models per property, although an increased number of shadow models improves the results [SLCE23]. This is in contrast with training thousands of shadow models, which is needed to carry out the original PIA proposals [AMS$^+$15, GWY$^+$18].

7. Note that this is a simplified explanation of the attack. For more details, refer to [SLCE23, Section II-B].

2.4.4 Threat Models for Distributed Training

In distributed ML training settings, model training always entails communication with other parties. In FL, each client communicates with the aggregator – while both the clients and the aggregator could be controlled by an adversary. Note that in the plain FL protocol, the aggregator receives model updates from the clients in the clear, which means that the aggregator has white-box access to individual client model parameters.

There are two main threat models to consider in distributed settings: In an *honest-but-curious* (or *passive*) setting, infiltrated parties follow the protocol but try to gain as much information as possible. This means that the training process is executed as expected, but the adversary collects all information available and can run further analyses. In contrast, the *malicious* (or *active*) setting allows corrupted parties to deviate from the protocol. Hence infiltrated clients could send arbitrary updates to the aggregator and the aggregator could send arbitrary model updates to the clients, allowing for sophisticated attacks.

In this dissertation (specifically Chapter 6), the focus is primarily on the anonymity of FL clients: Thus, a corrupted aggregator is considered, who tries to link inferred information from model updates to individual clients as illustrated in Figure 2.7. The protocol presented in Chapter 6 is tailored for the honest-but-curious setting, although an extension to the malicious setting is discussed.

2.4.5 Attacks on Distributed Training Processes

Since the aggregator has full white-box access to the clients' model updates in plain FL, all attacks on trained models which have been mentioned above could be carried out by an infiltrated aggregator on the received client updates. Furthermore, the aggregator is aware of which client has submitted each update. Together with the ability to compare the results of attacks on different individual contributions, this provides powerful adversarial capabilities and enables precise, fine-grained attacks [LYY20].

In the distributed setting, poisoning attacks are a natural threat, since an adversary controls part of the training data if a client has been infiltrated. While still considering this kind of attack as out of scope, the interested reader can be pointed to [BCMC19].

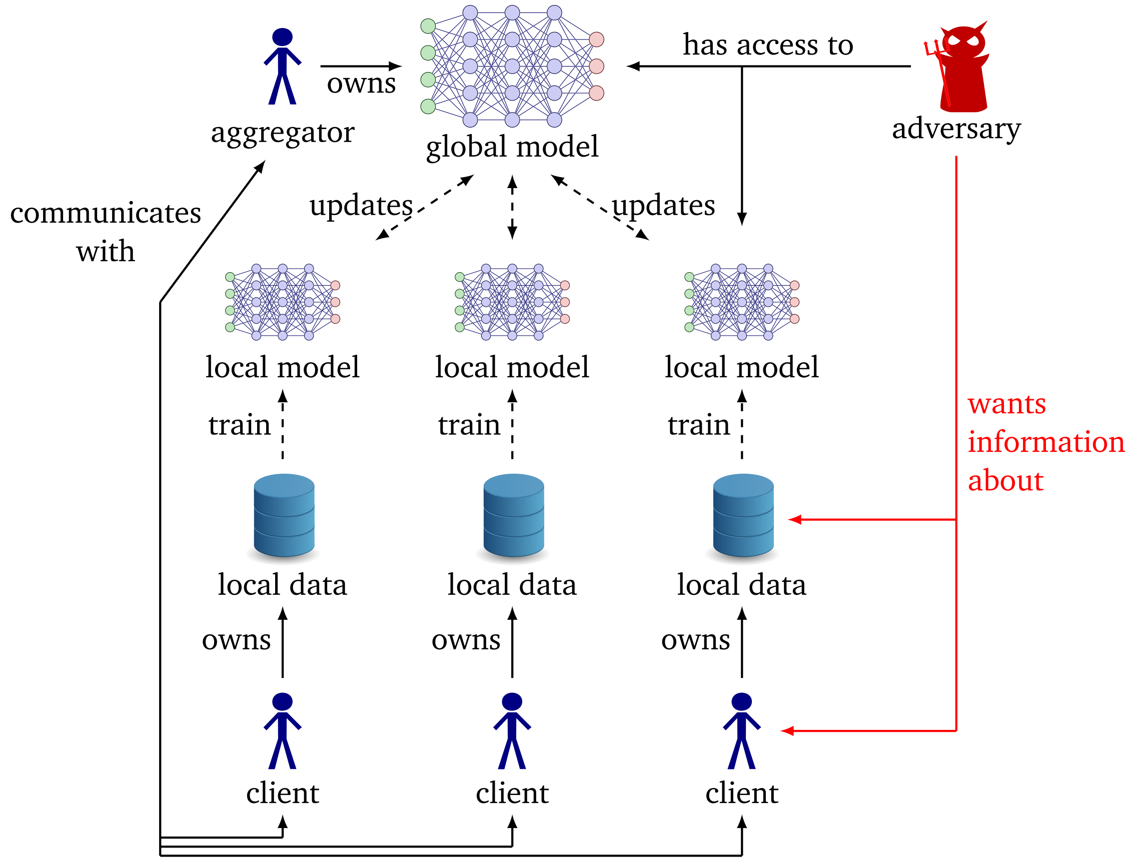


Figure 2.7: Threat model for FL with infiltrated aggregator. Dashed lines represent data flows, full lines represent possible actions. The adversary has access to *all* local models, arrows omitted for clarity. This graphic is inspired by [RG23] and includes content by Izaak Neutelings, licensed under CC BY-SA 4.0.

2.5 Preventing Information Leakage

Two of this dissertation's contributions (**C.3** and **C.5**) concern the mitigation of PIAs and the anonymity of FL clients. Related work concerning these topics is briefly summarized here.

2.5.1 Mitigating Property Inference

Differential privacy (DP) [Dwo06] is a privacy model aimed at reducing the impact of individual contributions to a computation. This is achieved by distorting the computation with randomness. In ML, or more specifically privacy-preserving machine learning (PPML), DP is often used to limit the influence of individual samples from a training data set on a trained model, thereby mitigating MIAs and other attacks targeting individuals [YGFJ18]. However, DP is not suitable as a defense strategy against PIAs, since they target *global* training data properties [AMS+15, SE22].

Due to the nature of ML models, removing sensitive attributes from training data does not help as well: Correlated information contained in other attributes is hard to rule out and can still lead to a leak of sensitive information [ZTO21]. In general, there are no established defense mechanisms against PIAs as of yet [SE22]. Hartmann et al. present a study on the reasons for property leakage and discuss directions for defense strategies on a theoretical level [HMP+23]. They suggest to use causal learning [TSN20], minimizing wrong inductive bias, e.g., by further optimizing the architecture of potential target models, and to increase the size of training data sets. Another recent proposal by Suri et al. is to change the distribution of training data itself, which hides the true distribution of a sensitive property [SLCE23]. However, this entails a major manipulation of the training data, just as the suggestion of node multiplicative transformations by Ganju et al. means a major change in model architecture [GWY+18]. Hence, all of these strategies violate first requirement formulated for **RQ2** in Section 1.2. In contrast, the strategies presented in Chapter 3 and Chapter 4 do not require changes in the model architecture or in the training data set, as they are variants of *adversarial learning*.

2.5.2 Adversarial Learning

The concept of adversarial learning is most commonly known for its application in training a generative adversarial network (GAN). A GAN [GPAM+14] consists of two main components, each modeled as an ANN: a generator G and a discriminator D. The concept is illustrated in Figure 2.8. While G tries to *generate* plausible data from latent noise, the task of D is to *discriminate* fake data produced by the generator from real data,

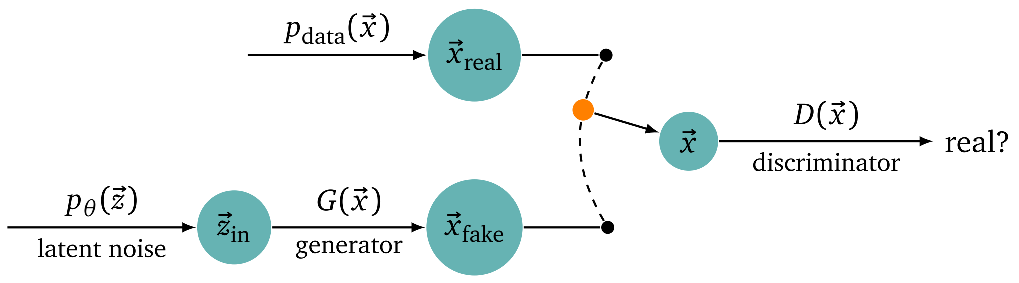


Figure 2.8: Schematic diagram of a generative adversarial network (GAN). By Janosh Riebesell, MIT licensed.

i.e., reveal the shortcomings of G. When D detects implausible results, i.e., correctly identifies fake data, a penalty is backpropagated into G. This is how the generator's weights are adapted to produce more realistic samples over time. Thus, the ultimate goal of a GAN is to train a generative model G which is able to create high-quality data, e.g., text or images, and fool the discriminator D. During its training process, G and D are trained in parallel. Therefore, the generator is forced to produce more realistic data samples, while the discriminator can adjust to the improving capabilities of the generator. So even though the discriminator is not necessarily used after training, it plays a vital part in training the generator – as an *adversary* to the generator.

Since its introduction in 2014 [GPAM+14], adversarial learning has been successfully applied to other ML domains: For instance, it has been shown to be the most reliable defense strategy [ACW18] against *adversarial examples* or model poisoning – attacks aimed at manipulating an ML model into making false predictions [SZS+13, GSS14]. More related to the topic of this dissertation, adversarial training has also been introduced as a successful mitigation strategy against black-box MIAs [NSH18b]. And as discussed in Section 4.4, it can also be used to achieve *fairness* in ML decisions [GRLD20].

The concept of adversarial learning is applied in both chapters Chapter 3 and Chapter 4 for defending against PIAs. Before introducing these strategies, methods for preventing information leakage on a different level need to be discussed: leakage *during* collaborative training processes, the topic of Chapter 6.

2.5.3 Secure Aggregation for Federated Learning

The state-of-the-art techniques to achieve anonymous training contributions for FL clients, in the sense that the aggregator cannot link individual model updates to specific clients, are based on *secure aggregation*. While there are secure aggregation strategies based on secure multi-party computation (SMPC) and homomorphic encryption (HE) as summarized in

Section 6.3, *masking* has the potential to be much less computationally expensive while being just as cryptographically secure [MÖJC23].

The intuition behind FL masking strategies is that clients can add a pseudo-random mask to their update before sending it to the aggregator. When the central party has received all updates, they can still be aggregated by computing a sum. Afterwards, the aggregated sum over all pseudo-random masks can be subtracted from the aggregated (masked) updates, such that the average (unmasked) update can be computed for the next training round. As long as the number of participating clients in a training round is high enough and the aggregator does not learn single masks, the aggregator has no insights into individual client contributions.

There are two main challenges to overcome in designing a good masking protocol: Computing (respectively serving) the aggregated mask to the aggregator and handling dropouts, i.e., clients who lose their connection to the aggregator within a training round. In Chapter 6, the newly proposed DealSecAgg protocol with a focus on communication and computation efficiency is compared to three other state-of-the-art masking strategies. As the theoretical analysis and experiments show, the other protocols all lack efficiency in overcoming either of the two challenges – being not resilient regarding dropouts or lacking scalability regarding mask aggregation in the first place, in contrast to DealSecAgg.

3 Lessons Learned: Defending Against Property Inference Attacks

Citation	Joshua Stock, Jens Wettlaufer, Daniel Demmler, and Hannes Federrath. (2023). Lessons Learned: Defending Against Property Inference Attacks. In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT)*. ISBN: 978-989-758-666-8, ISSN: 2184-7711, SciTePress, pages 312–323. DOI: 10.5220/0012049200003555
Venue	20th International Conference on Security and Cryptography (SECRYPT'23)
Ranking	**C** (CORE 2023), **B** (CORE 2021), **B4** (Qualis 2012), **B** (ERA 2010)
Status	Published. This contribution was presented at SECRYPT 2023: https://secrypt.scitevents.org/?y=2023.
Publication Type	Research Paper
Aim	The aim of this paper was to introduce and analyze a defense mechanism against white-box property inference attacks. For a better understanding of the presented defense mechanism's limitations, an analysis of the attack's functionality is also provided and discussed.
Methodology	The proposed adversarial learning defense mechanism is tested through a series of experiments. In order to measure the mechanism's efficiency, models were trained on three different data sets and attacked by the state-of-the-art white-box property inference attack. The attack success rates and the models' performance were measured for both the initial models and compared to models which were defended with the novel proposal. A variant of the defense mechanism where the approach was applied multiple times consecutively was also tested experimentally. The functionality of the attack was analyzed in experiments with a tool from the area of explainable artificial intelligence and a data visualization tool.

Continued on next page

Contribution	To the best of my knowledge, this is the first published work with the aim of developing a practical defense strategy against property inference attacks. Although the presented defense strategy is able to harden models against specific adversary instances, the defense does not generalize to a whole class of adversaries, i.e. it does not successfully hide a training data property from *all* potential property inference adversaries. The reasons for these limitations are thoroughly explored and discussed, in order to serve as a reference point for future research. Regarding the functionality of the attack, the two main findings of this work are that information about training data properties are both spread across large parts of the trained parameters of a model, and that they severely influence the shape of parameters, such that a data visualization tool is able to cluster model parameters with different training data properties apart. The code for the experiments of this paper is available open source [Sto25a].
Co-authors' contribution	This article was co-authored by Jens Wettlaufer who helped developing initial research ideas and assisted during the writing of the paper. He also designed Figures 3.1 and 3.2. Daniel Demmler assisted in revising and refining the paper. Hannes Federrath provided editorial guidance.

Abstract

This work investigates and evaluates defense strategies against *property inference attacks (PIAs)*, a privacy attack against machine learning models. While for other privacy attacks like membership inference, a lot of research on defense mechanisms has been published, this is the first work focusing on defending against PIAs. One of the mitigation strategies we test in this paper is a novel proposal called *property unlearning*. Extensive experiments show that while this technique is very effective when defending against specific adversaries, it is not able to generalize, i.e., protect against a whole class of PIAs. To investigate the reasons behind this limitation, we present the results of experiments with the explainable artificial intelligence tool LIME and the visualization technique t-SNE. These show how ubiquitous statistical properties of training data are in the parameters of a trained machine learning model. Hence, we develop the conjecture that post-training techniques like property unlearning might not suffice to provide the desirable generic protection against PIAs. We conclude with a discussion of different defense approaches, a summary of the lessons learned and directions for future work.

3.1 Introduction

The term *machine learning (ML)* describes a class of self-adapting algorithms which fit their behavior to initially presented training data. It has become a very popular approach to model, classify and recognize complex data such as images, speech and text. Due to the high availability of cheap computing power even in smartphones and embedded devices, the presence of ML algorithms has become a common sight in many real-world applications. At the same time, issues related to privacy, security, and fairness in ML are increasingly raised and investigated.

This work[1] focuses on ML with artificial neural network (ANN). After an ANN has been constructed, it can "learn" a specific task by processing big amounts of data in an initial training phase. During training, the connections between the network's nodes (or *neurons*) are modified such that the performance of the network regarding the specified task increases. After a successful training phase, the model, i.e., the network, is able to generalize, and thus enables precise predictions even for previously unseen data records. But while the model needs to extract meaningful properties from the training data to perform well in its dedicated task, it usually "remembers" more information than it needs to [SRS17]. This can be particularly problematic if training data contains private and sensitive information such as intellectual property or health data. The unwanted manifestation of such information, coupled with the possibility to retrieve

1. This is an abbreviated conference version. For the full paper, please refer to [SWDF22].

it, is called *privacy leakage*. In recent years, a new line of research has evolved around privacy leakage in ML models, which investigates privacy attacks and possible defense mechanisms [RG20].

In this paper, we focus on a specific privacy attack on ML models: the *property inference attack (PIA)*, sometimes also called *distribution inference* [AMS$^+$15, GWY$^+$18]. Given a trained ML model, PIAs aim at extracting statistical properties of its underlying training data set. The disclosure of such information may be unintended and thus dangerous as the following example scenarios show:

1. Computer networks of critical infrastructures have collaboratively trained a model on host data to detect anomalies. Here, a PIA could reveal the distribution of host types in the network to refine a malware attack.

2. Similarly, a model within a dating app has been trained on user data to predict good matches. Another competing app could use a PIA to disclose properties of the customer data to improve its service, e.g., the age distribution, to target advertisements more precisely.

If such models are published or leaked to the public on other channels, PIAs can reveal secrets of their training data. These secrets do not need to be in obvious correlation to the actual model task, like the property *host type* in the anomaly detection model of example 1.

Contributions

To the best of our knowledge, we are the first to evaluate defense strategies against PIAs, such as a novel approach called *property unlearning*. Our goal with property unlearning is to harden a readily trained ANN, further called *target model*, against PIAs, i.e., against the adversarial extraction of one or more predefined statistical properties in the training data set of a target model. The idea is to deliberately prune chosen properties from a target model, while keeping its utility as high as possible, thus protecting the privacy of the data set used for training.

Property unlearning is designed for the white-box attack scenario, where the adversary has full access to the internal parameters of a target model which are learned during the training phase. We have conducted thorough experiments which show that (a) property unlearning allows to harden ANNs against a specific PI attacker with small utility loss but (b) it is not possible to use the approach to completely prune a property from a trained model, i.e., to defend against *all* PI attackers for a chosen property in a generic way.

Consequently, we have conducted further experiments with the XAI tool LIME [RSG16] and the visualization framework t-SNE [VdMH08]. Both provide evidence for the conjecture that properties are ubiquitous in the trained weights of an ANN, such that complete pruning of a property from a trained ANN is not possible without greatly limiting its utility.

In the full version of this paper, we additionally investigate the impact of simple training data preprocessing steps such as adding Gaussian noise to images of a training data set on the success rate of PIAs. This is meant as an inspiration for possible alternatives to techniques such as differential privacy, which has been established as a de-facto standard against many privacy attacks with the exception of PIAs [RG20, SKMP22].

Organization of this paper

Section 3.2 briefly explains ANNs, ML privacy attacks, our threat model and PIAs. Section 3.3 deals with an overview of related work. Our defense strategy property unlearning is presented in Section 3.4. Section 3.5 describes our property unlearning experiments, including our findings regarding its limitations. We further experimentally explore the reasons for these limitations via the XAI tool LIME and t-SNE visualization in Section 3.6. We summarize and discuss our findings in Section 3.7. Directions for future work are provided in Section 3.8, and Section 3.9 concludes this paper.

3.2 Background

Notation We denote the set of integers $[k] = \{1,\dots,k\}$. Properties of a data set are denoted as blackboard bold, e.g., $\mathbb{A}$ and $\mathbb{B}$. Replacing the property-subscript with an $*$, we reference all possible data sets DS, e.g., DS_* means both $DS_\mathbb{A}$ and $DS_\mathbb{B}$. An *absolute* increase of x percent points is denoted as $+x$%P.

3.2.1 Artificial Neural Networks

An artificial neural network (ANN) consists of interconnected neurons, organized in multiple layers. Inputs are propagated through the network layer by layer. For this, each neuron has an associated *weight* factor w and a *bias* term b. A (usually non-linear) activation function σ computes each neuron's output on a given input, specifically for a neuron n and input x: $n = \sigma(w \cdot x + b)$

Prior to training an ANN, all neurons are individually initialized with random weights and biases (also called *parameters*). Utilizing a labeled training data set in an iterative *training* process, e.g., batch-wise backpropagation, these parameters are tuned such that the network predicts the associated label to its given input. The speed of this tuning process, respectively its magnitude per iteration, is controlled by the *learning rate*. The higher the learning rate, the more the parameters are adapted in each round.

3.2.2 Machine Learning Privacy Attacks

In general, privacy attacks against ML models extract information about training data of a target model $\mathscr{M}$ or the target model itself from its trained parameters. Some attacks, like membership inference [SSSS17] extract information about a single record from a ML model. Other attacks try to recover the model itself [PMG+17] or to recover the training data set or parts of it [FJR15]. In contrast, this paper focuses on *property inference attacks (PIAs)*, which reveal statistical properties of the entire training data set. This is not to be confused with *attribute inference attacks*, e.g., [SS20], which enable the adversarial recovery of sensitive attributes for *individual* data records from the training data set.

3.2.3 Threat model

In the remainder of this paper, the following threat model is assumed: A model owner has trained and shared the model of an ANN. The owner wishes to keep their training data and its property $\mathbb{A}$ or $\mathbb{B}$ (a statistical property of the training data) secret. An example may be a company that has trained a model on its customer data and does not want to disclose any demographic information about their customers. If an attacker gets access to this model, they can perform a PIA and reconstruct the demographics of its training data, breaching the desired privacy. In another scenario, an attacker might want to gather information about a computer network before launching a malware attack. Such networks are often monitored by intrusion detection systems (IDS), which have been trained on network traffic to detect unusual behavior. Having access to this IDS model, the attacker could infer the OS most computers are running on in the system, or even detect specific vulnerabilities in the network, as demonstrated in [GWY+18].

We assume that the attacker has full *white-box access* to the target model $\mathscr{M}$. This means that the attacker can access all parameters and some hyperparameters of $\mathscr{M}$: The adversary has a complete overview of the ANN architecture and can access the values of all weights and biases, as well as other useful hyperparameters of $\mathscr{M}$ such as the batch size during training, the learning rate and the number of training epochs. This helps the adversary to tailor their shadow models (see Section 3.2.4) as close to the target model as possible. In contrast, an adversary in a *black-box scenario* typically has oracle-access to the target model $\mathscr{M}$, allowing only to send queries to $\mathscr{M}$ and to analyze the corresponding results, i.e., the classification of a data instance.

As assumed in previous defenses against privacy attacks [NSH18, SM21, TMS⁺21], the attacker can access parts of the target model's training data, or knows a distribution of the training data, but cannot access the whole training data set. Information about the training data may also be reconstructed like in [SSSS17], which is just as effective for privacy attacks [LWH⁺22].

3.2.4 Property Inference Attacks (PIAs)

[AMS⁺15] were the first to introduce PIAs, with a focus on hidden Markov models and support vector machines. In this paper, we refer to the state-of-the-art PIA approach by [GWY⁺18] who have adapted the attack to fully connected neural networks, a popular sub-type of ANNs. In a typical PIA scenario, an adversary has access to a trained ML model called *target model* $\mathscr{M}$, but not its training data. By using the model at inference time, a PIA enables the adversary to deduce information about the training data which the model has learned. Since the adversary's tool for the attack is a ML model itself, we call it *adversarial meta classifier* $\mathscr{A}$. Thus, the adversary attacks the target model $\mathscr{M}$ by utilizing the meta classifier $\mathscr{A}$ to extract a property from its training data.

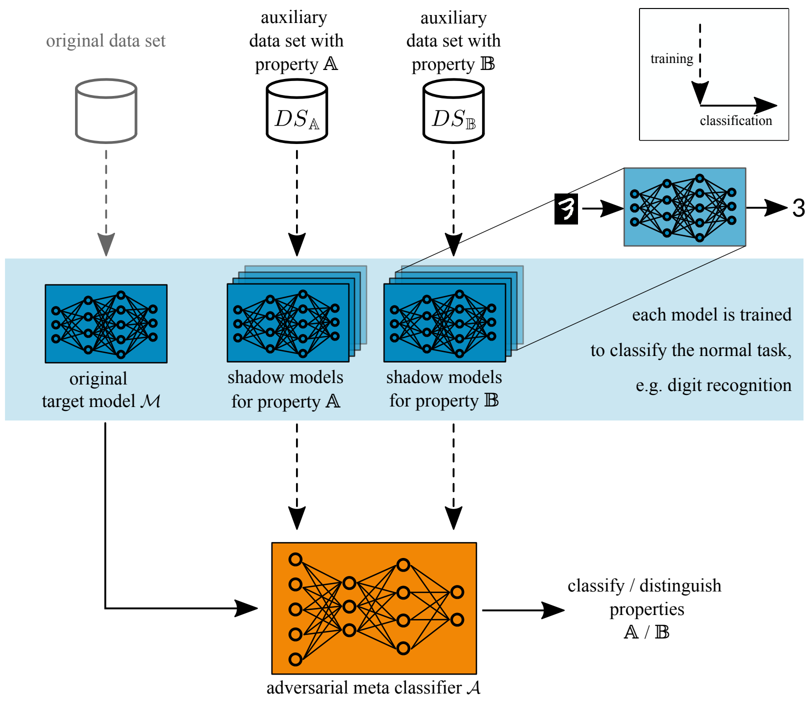
A PIA typically involves the following steps [GWY⁺18], see also Figure 3.1:

1. Define (at least) two global properties about the target model's training data set, e.g., $\mathbb{A}$ and $\mathbb{B}$. A successful PIA will show which property is true or more likely for the training data set of the given target model.

2. For each defined property, create an *auxiliary data set* DS_*, i.e., $DS_\mathbb{A}$ and $DS_\mathbb{B}$. Each auxiliary data set fulfills the respective property.

3. Train multiple *shadow models* on each auxiliary data set DS_*. Shadow models have the same architecture as the target model. Due to the randomized nature of ML training algorithms the weights and biases of every model have different initial values.

4. After training the shadow models, use their resulting parameters (weights and biases) to train the adversarial meta classifier $\mathscr{A}$. During this training, the meta classifier $\mathscr{A}$ learns to distinguish the parameters of target models that have been trained on data sets with property $\mathbb{A}$ and data sets with property $\mathbb{B}$, respectively. As a result, $\mathscr{A}$ is able to determine which of the properties $\mathbb{A}$ or $\mathbb{B}$ is more likely to be true for the training data of a given target model.



original data set
auxiliary data set with property $\mathbb{A}$
auxiliary data set with property $\mathbb{B}$
$DS_{\mathbb{A}}$
$DS_{\mathbb{B}}$
training
classification
3
each model is trained to classify the normal task, e.g. digit recognition
original target model $\mathcal{M}$
shadow models for property $\mathbb{A}$
shadow models for property $\mathbb{B}$
classify / distinguish properties $\mathbb{A}$ / $\mathbb{B}$
adversarial meta classifier $\mathcal{A}$

Figure 3.1: Property inference attack (PIA).

For example, suppose the task of a target model $\mathscr{M}$ is smile prediction with 50 000 pictures of people with different facial expressions as training data. For a PIA, the adversary defines two properties $\mathbb{A}$ and $\mathbb{B}$ about the target model's training data set, e.g.,

$$\mathbb{A}: \text{proportion of male:female data instances } 0.7{:}0.3$$
$$\mathbb{B}: \text{male and female instances are equally present.}$$

Given $\mathscr{M}$, the task of the adversary is to decide which property describes $\mathscr{M}$'s training data set more accurately. As mentioned in step 2., the adversary first needs to create two auxiliary data sets $DS_{\mathbb{A}}$ and $DS_{\mathbb{B}}$, with the male:female ratios as described in the properties above. After training shadow models on the auxiliary data sets, the adversary uses the trained weights and biases of the shadow models to train the adversarial meta classifier $\mathscr{A}$, which is ready for the adversarial task after its training.

The meta classifier can also be easily extended to more than two properties: For k properties, the adversary needs k auxiliary training data sets, trains shadow models in k groups and constructs $\mathscr{A}$ as a classifier with k outputs instead of two.

3.3 Related Work

This section briefly summarizes related work in the area of ML privacy attacks and defenses.

PIA defense strategies Effective universal defense mechanisms against PIAs have not been discovered yet [RG20]. Differential privacy [DMNS06] is a promising approach against other privacy attacks like membership inference [RG20, SKMP22]. However, it only slightly decreases the success rate of PIAs, since it merely limits the impact of each single input, but does not influence the presence of general properties in the training data set [AMS$^+$15, LWH$^+$22, ZTO21].
[GWY$^+$18] propose *node multiplicative transformations* as another defense strategy. As long as an ANN uses ReLU or LeakyReLU as an activation function, it is possible to multiply the parameters of one layer by some constant and dividing the constants connecting it to the next layer by the same value without changing the result. Although they claim that this might be effective, this strategy is limited to ReLU and LeakyReLU activation functions and requires changes in the model architecture. In contrast, the approaches we test in this paper do not require any changes to the target model and do not require specific activation functions.

Other PIA versions [MSDCS19] explore PIAs in the context of collaborative learning: Herein, the adversary is a legitimate party in a collaborative setting, where participants jointly train a ML model via exchanging model updates – without sharing their local and private data. The authors present an *active* and a *passive* method to infer a property of the training data of another participant by analyzing the shared model updates of other participants.

Focusing on a black-box scenario, [ZTO21] study both single- and multi-party PIAs for tabular, text and graph data sets. While their attack does not need access to the parameters of a target model, several hundreds of queries to the target model are needed for the attack to be successful.

An advanced PIA by [MGC22] introduces *poisoning* as a way to ease the attack in a black-box scenario. This requires the adversary to control parts of the training data. In this adversarial training data set, the label of data points with a target property $\mathbb{A}$ are changed to an arbitrary label l. After training, the distribution of a target property can then be inferred by evaluating multiple queries to the target model – loosely summarized, the more often the label l is predicted, the larger the portion of samples with property $\mathbb{A}$ is in the training data set.

[SS20] propose a very similar attack to property inference, which we call attribute inference: They assume a ML target model which is partly evaluated *on-premise* and partly *in the cloud*. Their attribute inference attack reveals properties of a single data

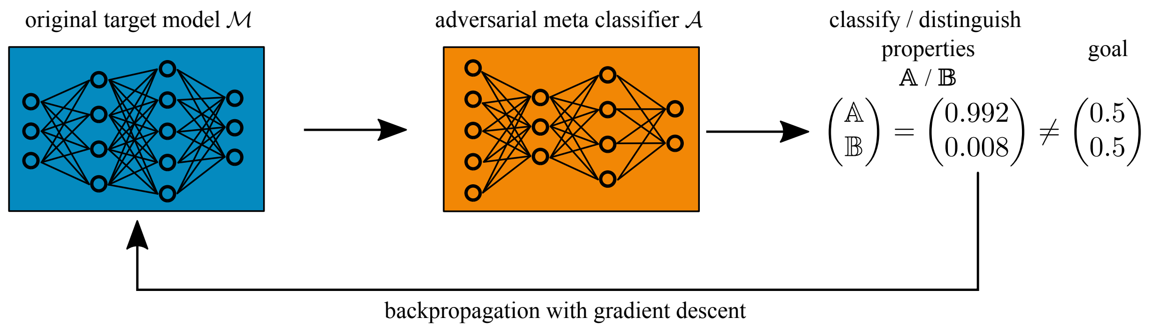

original target model $\mathcal{M}$
adversarial meta classifier $\mathcal{A}$
classify / distinguish properties $\mathbb{A}$ / $\mathbb{B}$
goal
$\begin{pmatrix}\mathbb{A}\\ \mathbb{B}\end{pmatrix} = \begin{pmatrix}0.992\\ 0.008\end{pmatrix} \neq \begin{pmatrix}0.5\\ 0.5\end{pmatrix}$
backpropagation with gradient descent

Figure 3.2: Property unlearning as a defense strategy against PIAs.

instance, e.g., whether a person wears glasses on a photo during the inference phase. In contrast, we focus on PIAs which reveal global properties about a whole training data set.

3.4 Property Unlearning

In this section we elaborate on our novel defense strategy against PIAs, which we call *property unlearning*. An overview of the approach is given in Figure 3.2.

As a prerequisite, an adversarial classifier $\mathscr{A}$ needs to be constructed. This is achieved as described in Section 3.2.4: constructing one auxiliary data set DS for each property $\mathbb{A}$ and $\mathbb{B}$, and training a set of *shadow models* for each property with the corresponding data sets $DS_{\mathbb{A}}$ and $DS_{\mathbb{B}}$. Note that when creating an adversary as a preparation for protecting one's own model, the auxiliary data sets $DS_{\mathbb{A}}$ and $DS_{\mathbb{B}}$ can trivially be subsets of the original training data of the target model, since the model owner has access to the full training data set. This yields a strong adversarial accuracy as opposed to an outside adversary who might need to approximate or extract this training data first. The same holds for white-box access to the model, which is straightforward for the owner of a model. Hence, the training of a reasonably good adversarial meta classifier $\mathscr{A}$ ($> 99\%$ accuracy) as a first step of property unlearning is easily achievable for the model owner (see Section 3.5). As a second prerequisite, the target model $\mathscr{M}$, which the owner wants to protect, also needs to be fully trained with the original training data set – having either property $\mathbb{A}$ or $\mathbb{B}$.

To *unlearn* the property from $\mathscr{M}$, we use backpropagation. As in the regular training process, the parameters of the target model $\mathscr{M}$ are modified by calculating and applying gradients. But different from original training, property unlearning does not optimize

$\mathscr{M}$ towards better classification accuracy. Instead, the goal is to disable the adversary $\mathscr{A}$ from extracting the property $\mathbb{A}$ or $\mathbb{B}$ from $\mathscr{M}$ while keeping its accuracy high.

In practice, the output of the adversarial meta-classifier $\mathscr{A}$ is a vector of length 2 (or: number of properties k) which sums up to 1. Each value of the vector corresponds to the predicted probability of a property. As an example, the output $[0.923, 0.077]$ means that the adversary $\mathscr{A}$ is 92.3% confident that $\mathscr{M}$ has property $\mathbb{A}$, and only 7.7% to have property $\mathbb{B}$. Thus, property unlearning aims to disable the adversary from making a meaningful statement about $\mathscr{M}$, i.e., an adversary output of $[0.5, 0.5]$ is pursued – or more generally $[\frac{1}{k}, \dots, \frac{1}{k}]$ for k properties.

Algorithm 1 shows pseudocode for the property unlearning algorithm. The termination

Algorithm 1 Property unlearning for a target model $\mathscr{M}$, using property inference adversary $\mathscr{A}$, initial learning rate lr, and set of properties $P = \{\mathbb{A}, \mathbb{B}, \dots\}$

```
 1: procedure PropUnlearning($\mathscr{M}$, $\mathscr{A}$, $lr$, $P$)
 2:     $k \leftarrow |P|$                              ▷ number of properties (default 2)
 3:     $Y \leftarrow \mathscr{A}(\mathscr{M})$                          ▷ original adv. output $|Y| = k$
 4:     let $i \in [k]$
 5:     while $\exists i : Y_i \gg \frac{1}{k}$ or $Y_i \ll \frac{1}{k}$ do
 6:         $g \leftarrow$ gradients for $\mathscr{M}$ s.t. $\forall i : Y_i \to \frac{1}{k}$
 7:         $\mathscr{M}' \leftarrow$ apply gradients $g$ on $\mathscr{M}$ with $lr$
 8:         $Y' \leftarrow \mathscr{A}(M')$                       ▷ update adversarial output
 9:         if AdvUtlt($Y'$) < AdvUtlt($Y$) then
10:             $\mathscr{M}, Y \leftarrow \mathscr{M}', Y'$
11:         else
12:             $lr \leftarrow lr/2$                          ▷ retry with decreased $lr$
13:         end if
14:     end while
15:     return $\mathscr{M}$
16: end procedure
17: function AdvUtlt(adv. output vector $Y$)
18:     $k \leftarrow |Y|$                              ▷ number of properties (default 2)
19:     return $\max_{i \in [k]}(|Y_i - \frac{1}{k}|)$                   ▷ biggest diff. to $\frac{1}{k}$
20: end function
```

condition for the while-loop in line 5 addresses the ability of the adversary $\mathscr{A}$: As long as $\mathscr{A}$ is significantly more confident for one of the properties, the algorithm needs to continue. After calculating the gradients g automatically via TensorFlow's backtracking algorithm in line 6, the actual unlearning happens in line 7. Here, the gradients are applied on the parameters of model $\mathscr{M}$, nudging them to be less property-revealing.

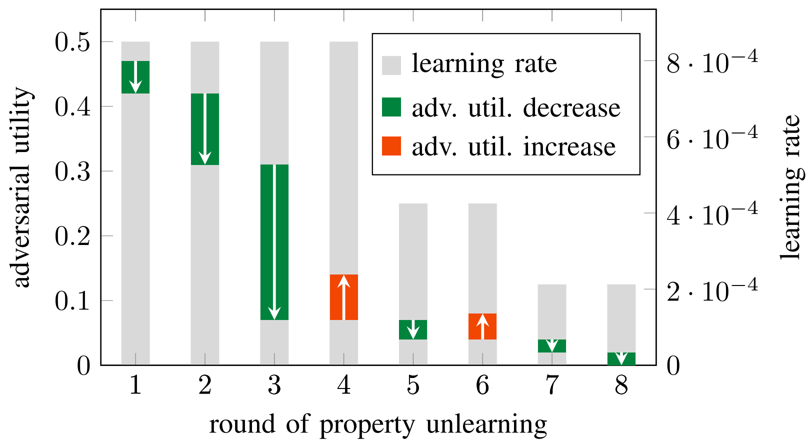


Figure 3.3: A visualized example of the decreasing adversarial utility during property unlearning with *one* adversary for a single target model $\mathcal{M}$. In each round, the adversarial utility of $\mathcal{M}$ either decreases further towards the goal of 0 (green bar), or the unlearning round is repeated with a smaller learning rate (after a red bar). The final result of round 8 is a completely *unlearned* target model $\mathcal{M}$ with an adversarial utility close to 0, see Algorithm 1.

As described in Section 3.2.1, the learning rate controls how much the gradients influence a single step. If the parameters have been changed *too much*, the current $\mathcal{M}'$ gets discarded and the gradients are reapplied with half the learning rate (see line 12 and visualization in Figure 3.3). Reducing the learning rate to its half has yielded the most promising results in our experiments.

The effect of property unlearning in between rounds of the algorithm is measured by the *adversarial utility*, see lines 17–20. We calculate the adversarial utility by analyzing the adversary output Y. Recall that Y is a vector with k entries, with each entry Y_i representing the adversarially estimated probability that the underlying training data set of the target model $\mathcal{M}$ has property i. The adversarial utility is defined by the largest absolute difference of an entry Y_i to $\frac{1}{k}$ (see line 19). Remember that the goal of property unlearning is to nudge the parameters of $\mathcal{M}$ such that the output of the adversary is close to $\frac{1}{k}$ for all k entries in the output vector Y. The condition in line 9 therefore checks whether the last parameter update from $\mathcal{M}$ to $\mathcal{M}'$ was useful, i.e., whether the adversarial utility has decreased. Only if this is the case, the algorithm gets closer to the property unlearning goal. Otherwise, the last update in $\mathcal{M}'$ is discarded and the next attempt is launched with a lower learning rate. A visualization of an exemplary run is given in Figure 3.3.

Experiment	Data set	Size	Target Property	$\lvert DS_*\rvert$	Shadow model accuracy	Init. PIA accuracy
$\mathscr{E}_{MNIST}$	[LBBH98]	70K	Gaussian noise	12K	88.3–94.5%	100%
$\mathscr{E}_{Census}$	*Census Income*	48K	gender distrib.	15K	84.7%	99.3%
$\mathscr{E}_{UTK}$	[ZSQ17]	23K	race distrib.	10K	88.0–88.3%	99.8%

Table 3.2: The data sets used for the experiments. init.=initial, distrib.=distribution

3.5 Property Unlearning experiments

To test property unlearning in practice, we have conducted extensive experiments with different data sets.

Adversarial property inference classifier As described in Section 3.2.4, we use the attack approach by [GWY+18]. This means that each instance of an adversary $\mathscr{A}$ is an ANN itself, made up of multiple sub-networks ϕ and another sub-network ρ. Per data set, we train one such adversarial meta classifier $\mathscr{A}$, which is able to extract the respective properties $\mathbb{A}$ and $\mathbb{B}$ from a given target model.

Depending on the number of neurons in a layer of the target model, our sub-NNs ϕ consist of 1–3 layers of dense-neurons, containing 4–128 neurons each. In the adversarial meta classifier $\mathscr{A}$, the number of layers and number of neurons within the layers are proportionate to the input size, i.e., the number of neurons in the layer of the target model. These numbers are evaluated experimentally, such that the meta classifiers perform well, but do not offer more capacity than needed (which would encourage overfitting).

Our sub-network ρ of $\mathscr{A}$ consists of 2–3 dense-layers with 2–16 dense-neurons each. In our experiments the output layer always contains two neurons, one for each property $\mathbb{A}$ and $\mathbb{B}$. For each of the three data sets in the next section, we apply the following steps to prepare for property unlearning:

- Design appropriate target model $\mathscr{M}$ for task.
- Extract two auxiliary data sets $DS_{\mathbb{A}}$ and $DS_{\mathbb{B}}$ for each property $\mathbb{A}$ and $\mathbb{B}$.
- Use each $DS_{\mathbb{A}}$ and $DS_{\mathbb{B}}$ as training data for 2000 shadow models. Shadow models have the same architecture as the target model $\mathscr{M}$.
- Design and train an adversarial meta classifier $\mathscr{A}$ on parameters of shadow models.

This adversarial model $\mathscr{A}$ may then be employed in our property unlearning algorithm (see Section 3.4).

Data sets and network architectures We use three different data sets to evaluate our approach, as summarized in Table 3.2. For each data set and auxiliary data set DS_*, we train 2000 shadow models and 2000 target models. For faster training and a more realistic scenario, the auxiliary data sets DS_* are smaller. While the shadow models are used to train the adversaries $\mathscr{A}$, the target models $\mathscr{M}$ are the subjects of our experiments, i.e., we apply property unlearning on these target models and measure the resulting privacy-utility trade-off. The shadow models and target models share the same architecture per data set.

MNIST is a popular database of labeled handwritten digit images. As in [GWY+18], we distort all images with Gaussian noise (parameterized with mean = 35, sd = 10) in a copy of the database. We choose the property of having original pictures without noise ($\mathbb{A}_{MNIST}$) and pictures with noise ($\mathbb{B}_{MNIST}$). Our models for the MNIST classification task are ANNs with a preprocessing-layer to flatten the images, followed by a 128-neurons dense-layer and a 10-neuron dense-layer for the output.

Census is a tabular data set for income prediction. The PIA aims at extracting the ratio of male to female persons in the database, which is originally 2:1. The auxiliary data set for property $\mathbb{A}_{Census}$ $DS_{\mathbb{A}_{Census}}$ has a male:female ratio of 1:1, $DS_{\mathbb{B}_{Census}}$ the original ratio of 2:1. The architecture of the Census models consists of one 20-neurons dense-layer and a 2-neurons output dense-layer.

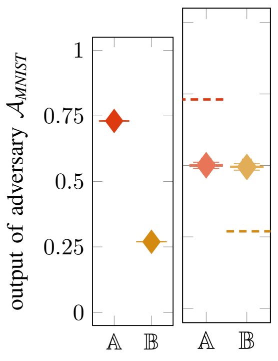
UTKFace contains over 23 000 facial images. We choose gender recognition as the task for the target models $\mathscr{M}$. Concerning our choice of properties, we create a data set consisting only of images with ethnicity *White* from the original data set for property $\mathbb{A}_{UTK}$. The data set for property $\mathbb{B}_{UTK}$ is comprised of images labeled with *Black*, *Asian*, *Indian*, and *Others*.

For UTKFace gender recognition, we use a convolutional neural network (CNN) architecture with three sequential combinations of convolutional, batch normalization, max-pooling and dropout layers, leading to one dense-layer with 2 neurons.

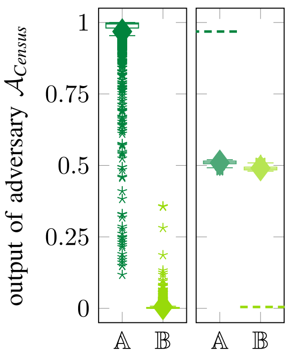
3.5.1 Experiment 1: Property unlearning

In this section we experimentally evaluate the performance of property unlearning to defend against a specific PIA adversary. For each of the data sets described above, we have trained 2000 test models in the same way we have created the shadow models. We refer to these test models as target models.

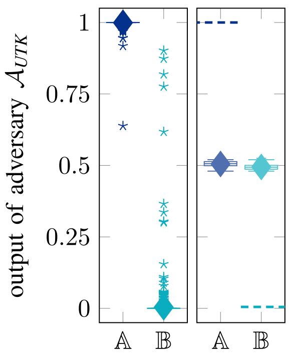
The figures in this section contain boxplot-graphs. Each boxplot consists of a box, which vertically spans the range between the first quartile Q_1 and the third quartile Q_3, i.e., the



(a) $\mathscr{E}_{MNIST}$

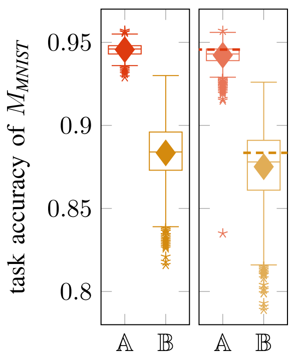


(b) $\mathscr{E}_{Census}$

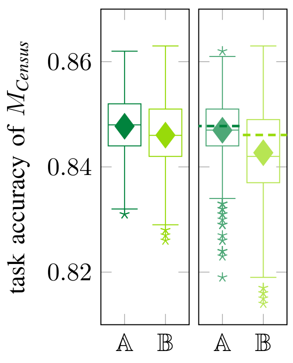


(c) $\mathscr{E}_{UTK}$

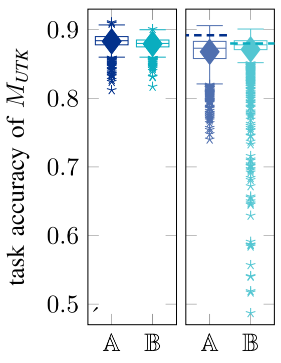
Figure 3.4: Each experiment before and after property unlearning, depicting the certainty of adversary $\mathscr{A}$ in classifying 𝔸 and 𝔹. The dashed lines represent the avg. accuracy *before* property unlearning was applied on 2000 target models.



(a) $\mathscr{E}_{MNIST}$



(b) $\mathscr{E}_{Census}$



(c) $\mathscr{E}_{UTK}$

Figure 3.5: Each experiment before and after property unlearning regarding the accuracy loss of the target models $\mathscr{M}$. The dashed lines represent the average accuracy values *before* property unlearning was applied on 2000 target models.

range between the median of the upper and lower half of the data set. The horizontal line in a box marks the median and the diamond marker indicates the average value.

MNIST For the MNIST experiment $\mathscr{E}_{MNIST}$, the adversary classifies the properties $\mathbb{A}$ and $\mathbb{B}$ with high certainty in all instances before unlearning, see Figure 3.4a. After unlearning, the adversary cannot infer the property of any of the MNIST target models $\mathscr{M}_{MNIST}$ – as intended. Meanwhile, the accuracy of the target models $\mathscr{M}_{MNIST}$ decreased slightly from an average of 94.6% by 0.4%P to 94.2% for models with property $\mathbb{A}$, respectively from 88.3% by 0.8%P to 87.5% for models with property $\mathbb{B}$ (see Figure 3.5a). Recall that property $\mathbb{B}$ was introduced by applying noise to the training data, hence the affected models perform worse in general.

Census Property unlearning was also successfully applied in the $\mathscr{E}_{Census}$ experiment to harden the target models $\mathscr{M}_{Census}$ against a PI adversary $\mathscr{A}_{Census}$, see Figure 3.4b. Note that the performance of $\mathscr{A}_{Census}$ is not ideal for property $\mathbb{A}$, classifying some of the instances incorrectly. However, 99.3% of the 2000 instances were classified correctly by the adversary *before* property unlearning. As desired, the output of $\mathscr{A}_{Census}$ is centered around 0.5 for both properties after property unlearning. The magnitude of the target models' accuracy loss is small, with an average drop of 0.1%P for property $\mathbb{A}$ (84.8% to 84.7%) and 0.3%P (84.6% to 84.3%) for property $\mathbb{B}$, see Figure 3.5b.

UTKFace In the $\mathscr{E}_{UTK}$ experiment, property unlearning could be successfully applied to all models (see Figure 3.4c) to harden the target models against PIAs. On average, the accuracy of the target models dropped by 1.3%P (from 88.2% to 86.9%) for models trained with the data set $DS_{\mathbb{A}}$ and by 0.1%P (from 87.9% to 87.8%) for target models trained with $DS_{\mathbb{B}}$, see Figure 3.5c. This yields an average accuracy drop of 0.8%P across the target models for both properties (from 88.1% to 87.3%).

3.5.2 Experiment 2: Iterative property unlearning

In the previous section, the results of Experiment 1 have shown that property unlearning can harden a target model $\mathscr{M}$ against a single PI adversary, i.e., a specific adversarial meta classifier $\mathscr{A}$ (see Figure 3.2). The setup of Experiment 2 aims to improve that by generalizing the unlearning. Therefore, the same target model $\mathscr{M}$ is unlearned iteratively against a range of different adversary instances $\mathscr{A}$ (see Figure 3.6). The results of our experiments are based on 200 target models. We unlearn each initial target model $\mathscr{M}^{(0)}$ iteratively for n different adversarial meta classifiers $\mathscr{A}_i$, where $n = 15$. After that, the resulting iteratively unlearned target model $\mathscr{M}^{(n)}$ is tested by another distinct adversarial meta classifier. To increase the significance of our results, we choose to test the resulting target model $\mathscr{M}^{(n)}$ with $m = 5$ additional distinct adversarial meta classifiers. Furthermore, we apply a 4-fold cross validation technique to this constellation

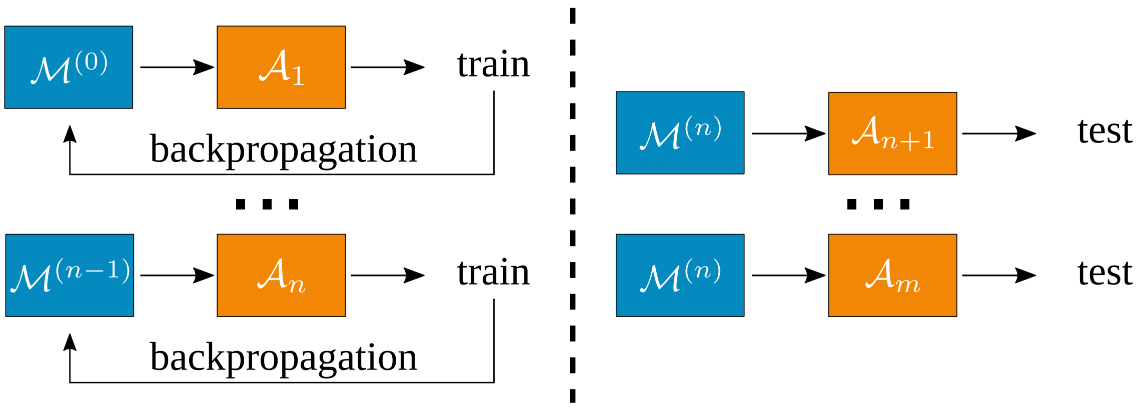


Figure 3.6: In reference to Figure 3.2, iterative property unlearning works by performing single property unlearning for n different adversarial meta classifier instances $\mathscr{A}$ iteratively on a target model $\mathscr{M}^{(0)}$. The resulting target model $\mathscr{M}^{(n)}$ is then evaluated by additional m instances of $\mathscr{A}$.

of in total 20 distinct adversarial classifiers. Finally, the results are plotted in boxplots similar to Experiment 1: Here, each boxplot is visualizing 200 (target models) $*$ 4 (folds) $*$ 5 (adversary outputs in a fold) $=$ 4000 data points.

The shadow models which were used to train the 20 adversaries $\mathscr{A}$ have been grouped such that the 5 testing adversaries' training set is disjunct from the training set of the 15 adversaries used for unlearning. The order of the 15 adversaries for unlearning has been chosen randomly for each of the 200 target models $\mathscr{M}$.

The overall results on the MNIST data set in Figure 3.7 show the iterative unlearning process for property $\mathbb{A}$ and $\mathbb{B}$. Each column on the x-axis represents an iteration step of the iterative unlearning procedure. On the y-axis, the prediction of the adversary regarding the corresponding property is plotted, which is ideal for property $\mathbb{A}$ and $\mathbb{B}$ to be 0 and 1, respectively. The goal of property unlearning is $y = 0.5$, such that the attacker is not able to distinguish the property. Clearly, the second column shows that after applying property unlearning once, a distinct adversary, i.e., not the adversary which was involved in the unlearning process, is still able to infer the correct property for most target models $\mathscr{M}$. The plots show that after about ten iterations of property unlearning, the average output of the 5 testing adversaries converges towards an average of prediction probability 0.5 (for both properties $\mathbb{A}$ and $\mathbb{B}$). While this could be misinterpreted as ultimately reaching the goal of property unlearning, we introduce Figure 3.8 which paints a more fine-grained picture of the last column of Figure 3.7. Here, each of the four plots contain five independent boxplots corresponding to the five distinct test adversaries in one fold of the cross validation process. Each boxplot presents the prediction results of one adversary for the 200 independently unlearned target models $\mathscr{M}_i^{(15)}$ of the experiment.

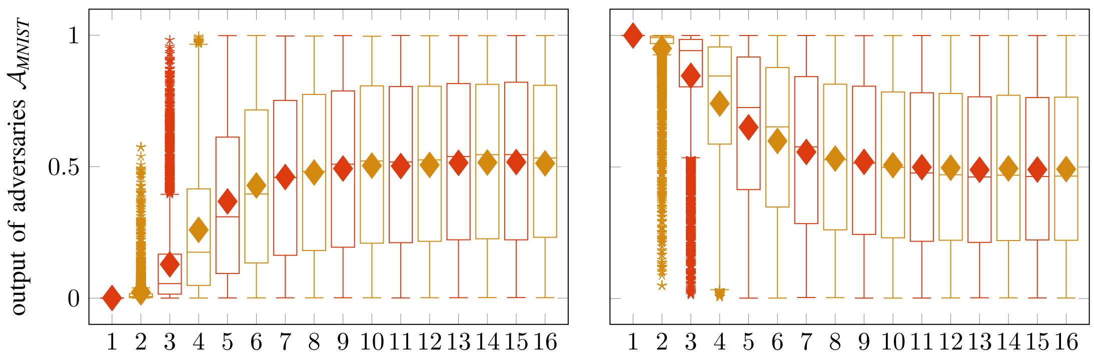


Figure 3.7: Results of iterative unlearning experiment for property $\mathbb{A}$ (left) and $\mathbb{B}$ (right). For each of the 200 target models $\mathscr{M}$, the predictions of all 5 testing adversaries are plotted along the y-axis before unlearning (first column) and after each unlearning iteration (other 15 columns).

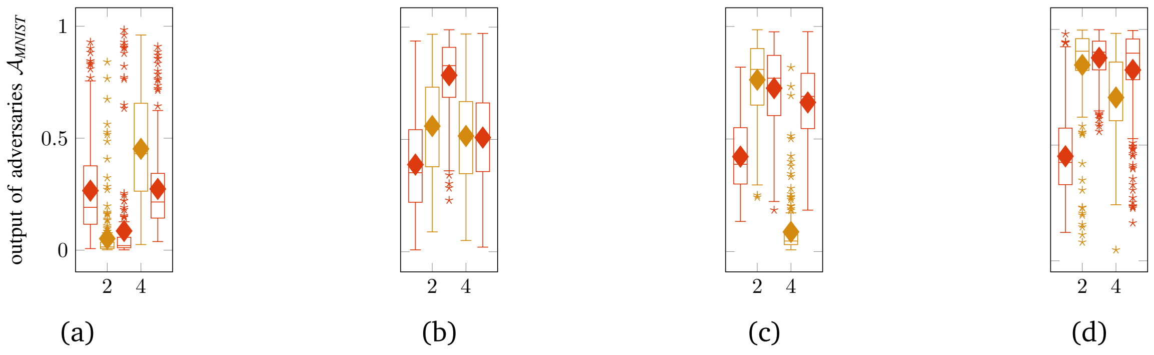


Figure 3.8: Individual adversary outputs after all 15 unlearning iterations for property $\mathbb{A}$ target models. Recall that before unlearning, all adversaries have correctly inferred property $\mathbb{A}$ by outputting $y = 0$.

While the plots of Figure 3.7 suggest that the adversaries' outputs are evenly spread across the interval $[0, 1]$ with both an average and median close to 0.5, Figure 3.8 shows that this is only true for the indistinct plot of all 4 experiments with 5 testing adversaries each. We want to point out three key observations:

1. Most adversaries do *not* have median outputs near 0.5 after 15 unlearning iterations.

2. For some adversary instances $\mathscr{A}$, target models have been "over-unlearned" by the 15 iterations with their output clearly nudged into opposite of their original output, e.g., adversary 3 in Figure 3.8d.

3. Most importantly, other adversaries are still correctly inferring the property for most or even all 200 target models with high confidence after the 15 unlearning iterations, e.g., the second adversary in Figure 3.8a.

3.5.3 Experiment discussion

Recall our goal for property unlearning: We want to harden target models in a generic way, such that arbitrary PI adversaries are not able to infer pre-specified properties after applying property unlearning.

Experiment 1 (single property unlearning) shows that property unlearning is very reliable to harden target models against specific adversaries. However, Experiment 2 (iterative property unlearning) indicates that single property unlearning fails to generalize, i.e., protect against all PI adversaries of the same class. This is shown in Experiment 2 by putting each target model through 15 iterations of property unlearning with one distinct adversary per iteration. After this, some adversaries are still able to infer the original properties of all target models (see third key observation in Section 3.5.2). This means that in the worst case, i.e., for the strongest adversaries, 15 iterations of property unlearning do not suffice – while for other (potentially weaker) adversaries, 15 or even less iterations are enough to harden the models against them. In conclusion, property unlearning does not meet our goal of being a generic defense strategy, i.e., protecting against a whole class of adversaries instead of a specific adversary.

3.6 Explaining PI Attacks

To explore the reasons behind this limitation of property unlearning, we use the XAI tool by [RSG16]: *Local interpretable model-agnostic explanations (LIME)* allows to analyze decisions of a black-box classifier by permuting the values of its input features. By observing their impact on the classifier's output, LIME generates a comprehensible ranking of the input features.

Recall that in the previous experiment (Section 3.5.2), we have seen that adapting the weights of a target model $\mathscr{M}$ s.t. an adversarial meta-classifier $\mathscr{A}_1$ cannot launch a successful PIA does not defend against another adversarial meta-classifier $\mathscr{A}_2$ trained for the same attack. Therefore, we use LIME to see whether different meta-classifiers $\mathscr{A}_1$ and $\mathscr{A}_2$ rely on the same weights of a target model $\mathscr{M}$ to infer $\mathbb{A}$ or $\mathbb{B}$.

For comprehensible results, we use LIME images. We convert the trained parameters of an MNIST target model $\mathscr{M}$ into a single-dimensional vector with length 101 770, so LIME can interpret them as an image. For segmentation, we use a dummy algorithm which treats each weight of $\mathscr{M}$ (resp. pixel) as a separate segment of the 'image'. This is necessary because unlike in an image, neighboring 'pixels' of $\mathscr{M}$'s weights do not necessarily have semantic meaning. For reproducible and comparable results, we have initialized all LIME instances with the same random seed.

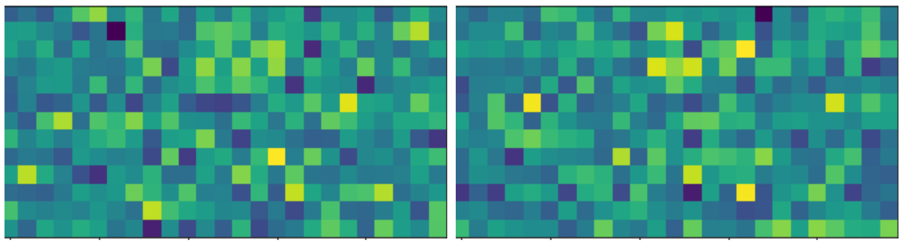


Figure 3.9: LIME produced, partial heat maps of different meta-classifier instances $\mathscr{A}_1$ and $\mathscr{A}_2$ for the same MNIST target model $\mathscr{M}$. Dark pixels represent parameters with high impact on the decision of $\mathscr{A}$, yellow pixels imply a low impact.

LIME Results We have instantiated LIME with two meta-classifiers $\mathscr{A}_1$ and $\mathscr{A}_2$ to explain their output for the same MNIST target model instance $\mathscr{M}$. The output of LIME is a heat map representing the weights and biases of $\mathscr{M}$, see Figure 3.9. For practical reasons, we have only visualized the first 784 pixels of the heat map and transformed them to a two-dimensional space. Although $\mathscr{A}_1$ and $\mathscr{A}_2$ are trained in the same way and with the same shadow models (see Section 3.5), the two heat maps for classifying the property of the same target model $\mathscr{M}$ in Figure 3.9 are clearly different: While some of $\mathscr{M}$'s weights have similar importance, i.e., the heat map pixels have a similar color, many weights have very different importance for the two adversarial meta classifiers $\mathscr{A}_1$ and $\mathscr{A}_2$.

To understand why meta-classifiers can rely on different parts of target model parameters to infer a training data property, we analyze the parameter differences induced by such properties on an abstract level. *t-SNE* [VdMH08]) is a form of dimensionality reduction which is useful for clustering and visualizing high-dimensional data sets. In particular, the algorithm needs no other input than the data set itself and some randomness.

In the t-SNE experiment, the input data set is comprised of the trained weights and biases of the shadow models. We apply this to the three data sets MNIST, Census and UTKFace. As before, we use 2000 shadow models (1000 with property $\mathbb{A}$ and 1000 with property $\mathbb{B}$). Our goal is revealing to which extend the trained parameters are influenced by a statistical property of the training data set. In particular, if the data agnostic approach t-SNE is able to cluster models with different properties apart, we can assume the influence of a property on model parameters to be significant.

t-SNE Results As depicted in Figure 3.10, t-SNE has produced a well defined clustering for the two image data sets MNIST and UTKFace: models trained with property $\mathbb{A}$ training data sets (yellow dots) are placed close to the center of the visualization, while property $\mathbb{B}$ models (purple dots) are mostly further from the center. This indicates that the properties, defined in Section 3.5 for MNIST and UTKFace, heavily influence the weights and biases of the trained models. In fact, without any additional information about the parameters or the properties of the underlying training data sets, t-SNE is able to distinguish the

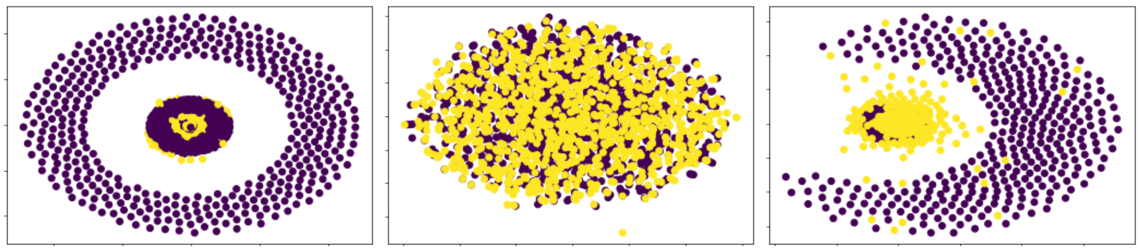


Figure 3.10: t-SNE visualization of MNIST (left), Census (center) and UTKFace (right) models. Each yellow dot represents a model with property $\mathbb{A}$, each purple dot a $\mathbb{B}$ model.

models by property with surprisingly high accuracy. Based on these results, one could construct a simple PI adversary $\mathscr{A}^{t\text{-}SNE}$ by measuring the euclidean distance ℓ of a target model from the center of the t-SNE clustering. If ℓ is below a certain threshold for a target model $\mathscr{M}$, $\mathscr{A}^{t\text{-}SNE}$ infers property $\mathbb{A}$, otherwise it infers property $\mathbb{B}$. For MNIST, $\mathscr{A}^{t\text{-}SNE}$ has 86.7% accuracy based on our experiment, while the UTKFace $\mathscr{A}^{t\text{-}SNE}$ has 72.0% accuracy. We stress that these two $\mathscr{A}^{t\text{-}SNE}$ are solely based on the t-SNE visualization of the model parameters, no training on shadow models is needed.

However for Census, t-SNE has not clustered models with different properties of their training data sets apart (see second visualization in Figure 3.10). In contrast to the other two data sets MNIST and UTKFace, Census is a tabular data set. It also may be that the properties defined in Section 3.5 have a smaller imminent impact on the weights and biases during training. We leave a more profound analysis of possible reasons for the different behavior of the t-SNE visualization on the three data sets for future work.

3.7 Discussion

We now discuss our results to yield insights for future research in the yet unexplored field of defending against PIAs.

Choosing the right defense approach We have introduced defense mechanisms at different stages of the ML pipeline. Both property unlearning experiments are positioned after the training and before its prediction phase, respectively its publication. In contrast, the preprocessing approach is applied prior to the training. Since most ML algorithms require several preprocessing steps, implementing a defense mechanism based on preprocessing training data could be easily adapted in real-world scenarios. At least for tabular data, our preprocessing experiments (see full paper [SWDF22]) have shown a good privacy-utility trade-off, especially the artificial data approach. Nevertheless, depending on the organization and application scenario of a ML model, a post-training approach like property unlearning might have its benefits as well. Further experiments could test the combination of both pre- and post-training approaches. Since both of them are not promising to provide the generic PIA defense we aimed for, we assume the combination of both does not significantly improve the defense. Instead, we suggest to focus further analyses on other approaches *during* the training, as laid out in Section 3.8.

Lessons learned With our cross-validation experiment in Section 3.5.2, we have shown how PI adversaries react to **property unlearning** in different ways. Some adversaries could still reliably infer training data properties after 15 property unlearning iterations, while other adversaries reliably inferred the *wrong* property after the same process. This shows that it is **hard to utilize a post-training technique** like property unlearning as a generic defense against a whole class of PI adversaries: After all, one needs to defend against the strongest possible adversary while simultaneously being careful not to introduce additional leakage by adapting the target model too much. Depending on the adversary instance, most of our target models clearly show one of these deficiencies after 15 rounds of property unlearning.

Our t-SNE experiment in Section 3.6 shows that at least for image data sets, statistical **properties of training data sets have a severe impact** on the trained parameters of a ML model. This is in line with the LIME experiment, which shows how two PI adversaries with the same objective focus on different parts of target model parameters. If a property is manifested in many areas of a model's parameters, PI adversaries can rely on different regions. This implies that completely pruning such properties from a target model after training is hard to impossible, without severely harming its utility.

3.8 Future work

Preprocessing training data We have not tested training data preprocessing in an *adaptive* environment yet, where the adversary would adapt to the preprocessing steps and retrain on shadow models with preprocessed training data as well. Intuitively, this would weaken the defense while costing the same utility in the target models. Additionally, as the technique with most potential for defending against PIAs for tabular data, the generation of artificial data could be further explored: One could adapt the synthesis algorithm s.t. statistical properties are arbitrarily modified in the generated data set. A similar goal is pursued in many bias prevention approaches in the area of fair ML.

Adapting the training process Another method from a similar area called *fair representation learning* is punishing the model when learning biased information by introducing a regularization term in the loss function during training, e.g., [CMJ$^+$19]. As a defense strategy against PIAs, one would need to introduce a loss term which expresses the current property manifestation within the model and causes the model to hide this information as well as possible. In theory, this would be a very efficient way to prevent the property from being embedded in the model parameters. Since it would be incorporated into the training process, the side effects on the utility of the target model should be low.

Post-training methods [LWH$^+$22] experiment with knowledge distillation (KD) as a defense against privacy attacks like membership inference. The idea is to decrease the number of neurons in an ANN in order to lower its memory capacity. Unfortunately, the authors do not consider PIAs – it would be interesting to see the impact of KD on their success rate.

3.9 Conclusion

In this paper, we performed the first extensive analysis on different defense strategies against white-box PIAs. This analysis includes a series of thorough experiments on *property unlearning*, a novel approach which we have developed as a dedicated PIA defense mechanism. Our experiments show the strengths of property unlearning when defending against a dedicated adversary instance and also highlight its limits, in particular its lacking ability to generalize. We elaborated on the reasons of this limitation and concluded with the conjecture that statistical properties of training data are deep-seated in the trained parameters of ML models. This allows PI adversaries to focus on different parts of the parameters when inferring such properties, but also opens up possibilities for much simpler attacks, as we have shown via t-SNE model parameter visualizations.

Apart from the *post-training* defense property unlearning, we have also tested different training data *preprocessing* methods (see full paper version [SWDF22]). Although most of them were not directly targeted at the sensitive property of the training data, some methods have shown promising results. In particular, we believe that generating a property-free, artificial data set based on the distribution of an original training data set could be a candidate for a PIA defense with very good privacy-utility tradeoff.

Acknowledgements

We wish to thank Anshuman Suri for valuable discussions and we are grateful to the anonymous reviewers of previous versions of this work for their feedback.

References

[AMS$^+$15] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *IJSN*, 2015.

[CMJ$^+$19] Elliot Creager, David Madras, Jörn-Henrik Jacobsen, Marissa Weis, Kevin Swersky, Toniann Pitassi, and Richard Zemel. Flexibly fair representation learning by disentanglement. In *ICML*, 2019.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*, 2015.

[GWY$^+$18] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *CCS*, 2018.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *IEEE*, 1998.

[LWH$^+$22] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. ML-Doctor: Holistic risk assessment of inference attacks against machine learning models. In *USENIX Security*, 2022.

[MGC22] Saeed Mahloujifar, Esha Ghosh, and Melissa Chase. Property inference from poisoning. In *S&P*, 2022.

[MSDCS19] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *S&P*, 2019.

[NSH18] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *CCS*, 2018.

[PMG$^+$17] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ASIACCS*, 2017.

[RG20] Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *Arxiv*, 2020.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you? explaining the predictions of any classifier. In *SIGKDD*, 2016.

[SKMP22] Anshuman Suri, Pallika Kanani, Virendra J Marathe, and Daniel W Peterson. Subject membership inference attacks in federated learning. *Arxiv*, 2022.

[SM21] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *USENIX Security*, 2021.

[SRS17] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *CCS*, 2017.

[SS20] Congzheng Song and Vitaly Shmatikov. Overlearning Reveals Sensitive Attributes. In *ICLR*, 2020.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *S&P*, 2017.

[SWDF22] Joshua Stock, Jens Wettlaufer, Daniel Demmler, and Hannes Federrath. Lessons learned: Defending against property inference attacks. *arXiv preprint arXiv:2205.08821*, 2022.

[TMS$^+$21] Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. Mitigating membership inference attacks by self-distillation through a novel ensemble architecture. *USENIX Sec.*, 2021.

[VdMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *JMLR*, 2008.

[ZSQ17] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *CVPR*, 2017.

[ZTO21] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of dataset properties in multi-party machine learning. In *USENIX Security*, 2021.

4 Property Inference as a Regression Problem: Attacks and Defense

Citation	Joshua Stock, Lucas Lange, Erhard Rahm, and Hannes Federrath. (2024). Property Inference as a Regression Problem: Attacks and Defense. In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT)*, ISBN: 978-989-758-709-2, ISSN: 2184-7711, SciTePress, pages 876–885. DOI: 10.5220/0012863800003767
Venue	21st International Conference on Security and Cryptography (SECRYPT'24)
Ranking	**C** (CORE 2023), **B** (CORE 2021), **B4** (Qualis 2012), **B** (ERA 2010)
Status	Published. This contribution was presented at SECRYPT 2024: https://secrypt.scitevents.org/?y=2024.
Publication Type	Research Paper
Aim	The aim of this paper was to explore property inference attacks as a regression problem and to develop an effective and practical defense strategy against them. The approaches developed should be tested and, if possible, compared with related work.
Methodology	A novel regression attack in the black-box threat model is implemented and evaluated in experiments with three different training data sets. A regression white-box attack is implemented as a benchmark for the black-box attack, such that their results can be compared in the experiments. A defense strategy for the black-box attack based on adversarial learning is proposed, implemented and tested in experiments. A review of related work with comparable approaches complements this paper.

Continued on next page

Contribution	The paper presents a novel regression property inference attack in the black-box threat model, as well as a white-box benchmark and a defense strategy for the black-box variant. In the experiments of this paper, the regression attack reaches R^2 test values between 0.60 and 0.86. The defense approach is highly effective, reducing the adversary's average test R^2 from 0.63 to 0.07. Regarding the performance of defended models, an accuracy drop of no more than 0.2 percentage points could be observed. The code for the experiments of this paper is available open source [Sto25b].
Co-authors' contribution	This article was co-authored by Lucas Lange who was a partner in developing research ideas and assisted during the writing of the paper. Erhard Rahm and Hannes Federrath provided editorial guidance.

Abstract

In contrast to privacy attacks focussing on individuals in a training dataset (e.g., membership inference), property inference attacks (PIAs) are aimed at extracting population-level properties from trained machine learning (ML) models. These sensitive properties are often based on ratios, such as the ratio of male to female records in a dataset. If a company has trained an ML model on customer data, a PIA could for example reveal the demographics of their customer base to a competitor, compromising a potential trade secret. For ratio-based properties, inferring over a continuous range using regression is more natural than classification. We therefore extend previous white-box and black-box attacks by modelling property inference as a regression problem. For the black-box attack we further reduce prior assumptions by using an arbitrary attack dataset, independent from a target model's training data. We conduct experiments on three datasets for both white-box and black-box scenarios, indicating promising adversary performances in each scenario with a test R^2 between 0.6 and 0.86. We then present a new defense mechanism based on adversarial training that successfully inhibits our black-box attacks. This mechanism proves to be effective in reducing the adversary's R^2 from 0.63 to 0.07 and induces practically no utility loss, with the accuracy of target models dropping by no more than 0.2 percentage points.

4.1 Introduction

Machine learning (ML) technologies are as present as never before and their advancement is significantly enhancing capabilities across multiple domains. However, this progress also introduces new challenges, particularly in the realm of data privacy. In healthcare and other user-centric areas, training large ML models requires ever increasing amounts of personal data to provide the desired outcomes. Dangers to privacy mostly stem from ML models being prone to leaking information about their training data under adversarial attacks [ARC19]. A famous example among these are membership inference attacks [SSSS17], deciding membership of individuals in training data, given a trained model. More recently property inference attacks (PIAs) [AMS+15] have emerged as a critical threat, where adversaries aim to infer sensitive properties from the training data of target models without direct access to the underlying data. Examples for such critical properties include the gender ratio (male to female) in a dataset or the status of security patches in log files used by an intrusion detection system [GWY+18]. These adversarial threats lead to an increasing body of research in privacy-preserving machine learning (PPML) that revolves around attacking and defending ML models in various scenarios [XBJ21].

The threat of PIAs is especially severe in white-box scenarios, where attackers have complete knowledge of the target model's architecture and parameters [AMS+15, GWY+18, SWDF23]. However unlike their white-box counterparts, black-box attacks do not require detailed knowledge of the model, reflecting more realistic adversarial scenarios against services that only query models for their outputs [ZTO21]. We introduce a novel method of training a black-box adversary for property inference, where we extend the conventional black-box PIAs to use an arbitrary attack dataset as input to the target model, which can be independent from the model's training data. By formulating PIAs as a regression problem, the adversary can freely extract the most likely distrubtion of a property in the original training data [SE22]. We show the feasibility of our black-box PIA in attacking Random Forest models and artificial neural networks (ANNs). To put our black-box results into context, we compare them to an adapted white-box attack by [GWY+18] that we run for the same scenarios. In all scenarios, we are able to obtain adversaries with a coefficient of determination (R^2) of at least 0.6, ranging up to 0.72 with black-box access and up to 0.86 with white-box access.

As a defense mechanism to our black-box PIA, we test the possiblity of directly including an adversarial loss in model training, which is inspired by [GRLD20] who showed the effectiveness for enhancing a model's fairness. The idea is to check the model's proneness to PIAs after each training round and to then accordingly influence the model's next updates to steer the model into hiding the original property distibution from its outputs. The model is instead trained to blind the adversary with a predefined target property value. We evaluate the resulting ability on mitigating the risks associated with black-box PIAs but also monitor the negative impact on model utility from impairing model training. Our results demonstrate that our mechanism effectively reduces black-box PIA success by an order of magnitude to an R^2 of 0.07, while only incurring a negligible utility loss of less than 0.2% accuracy.

We summarize our contributions as follows:

- We expand current PIAs for the black-box setting by enabling our meta-classifier to use an arbitrary attack dataset, which is indepedent from the target's original data distribution.
- For this meta-classifier, we frame our property inference as a regression problem that is able to extract a more realistic range of sensitive property distributions compared to the predefined distributions of a classification task.
- As a new defense mechanism we propose an adversarial training scheme that hides the actual property distribution from attacks by guiding the model during training to produce more balanced outputs, while preserving model utility. In contrast to defense schemes in related work, our method generalizes well, i.e., defends against a whole class of PIAs instead of defending against a single adversary instance.

For organizing this work, Section 4.2 provides an introduction to important related work on PIAs and possible defense mechanisms. Section 4.3 describes our methodology for conducting black-box PIAs and framing these attacks as a regression problem. Section 4.4 then outlines the proposed countermeasures for mitigating PIAs. In Section 4.5, we present our experimental setup, including the datasets used, the models evaluated, and the metrics for assessing the success of attack and defense. Section 4.6 discusses the results of our experiments, demonstrating the effectiveness and limitations of our approaches, and suggests future research directions. Finally, Section 4.7 concludes with a summary of our findings.

Performance metrics

Since we model the adversarial task as a regression problem, we measure the adversary's success with the *coefficient of determination (R^2)*. A dummy regressor always outputting the expected value of the trained labels would yield an R^2 of 0, while an ideal regressor's predictions would amount to an R^2 of 1.

In our evaluation, we use *boxplot graphs*, since they capture multiple characteristics of a distribution: the average is plotted as a diamond, while a line within the box is the mean of the distribution. The upper and lower line of the boxes capture the first and third quantile, while the whiskers below and above the box imply the range of values outside the two main quantiles. Outliers are plotted as points above or below the whiskers.

4.2 Related Work

The task of property inference, also known as distribution inference, is first introduced by [AMS+15] who describe attack patterns on traditional machine learning models. The current state-of-the-art white-box PIA by [GWY+18] is based on permutation-invariant networks and adapts earlier PIAs with a focus on fully connected ANNs. Their solution has since found successful adoption to other ML models and is the main focus for evaluating white-box PIAs [HMP+23, SE22]. In the same vein, we find the first approach for black-box attacks introduced by [ZTO21] to form the basis for the current implementations of black-box PIAs in these works. Their attack method uses a set of shadow models each trained on data exhibiting different distributions of the attacked property. They then train a meta-classifier that learns to predict the property of interest based on the shadow models' outputs on a query dataset. During execution, the attacker queries the target model with the attack query set, feeds the outputs to the meta-classifier, and obtains a prediction of the sensitive property in the data. The standard attack setting for PIAs is a binary classification over two predefined distribution values to infer to which of them the dataset property conforms. For a more realistic case of continuous distribution values like for the gender ratio, other works extend the inference task from binary to a larger set of possible distributions [ZTO21, ZCSZ22, SE22].

For defending against PIAs, [HMP$^+$23] try to evaluate the fundamental reasons for why models leak sensititve information about their distribution. They come to believe that three factors play an intervening role in enabling such exploits. As solutions they conversly suggest to reduce the model's memorization about the expected label given the features of adversary interest, optimizing model architecture against wrong inductive bias, and increasing the amount of training data. While these mitigations may help, they each come at a cost. Reducing memorization impacts model performance, optimizing model architecture requires prior assumptions, and collecting more training data is not always feasible.

Other methods for protecting against PIAs also have shown only limited effectiveness in reliably preventing their success. In contrast to membership inference attacks, differential privacy techniques are not effective at mitigating inference risks related to data distributions as tested by [AMS$^+$15]. This is because differential privacy focuses on obfuscating the contributions of individual records, whereas in the PIA setting, the adversary is able to uncover statistical properties of the underlying data distribution on a population level. Another approach by [ZTO21] tries to remove sensitive attributes from datasets, which has also proven ineffective, as the correlations between different attributes still prevails. [GWY$^+$18] propose a defense using scaling transformations applied in models with ReLU activations that hide their learned distributions in the internal weights. However, this technique does not offer any protection against black-box attacks. [ZCSZ22] suggest alterations to the training data with respect to the target property before training the model. This can be done in two ways: either by removing records or by adding new records. By this, the ratio of a property should reach a predetermined fixed value, such as 0.5, regardless of the original ratio. However, this strategy would either strive for acquiring new data or removing records from the dataset to balance the property ratio, leading to expensive data acquisition or a potential reduction in training data. In [SWDF23], they test property unlearning as a defense mechanism against white-box attacks. It uses an adversarial classifier to identify the model parameters that leak information about a property. They then use backpropagation to modify them into unlearning the property. Experiments show this can be effective against a specific white-box adversary, but has limitations in generalizing and can therefore not protecting against black-box or different white-box PIAs targeting a certain property. In summary, the currently available defences against property inference still have significant limitations and do not provide reliable protection across various attack scenarios.

To address the limitiations stemming from related work, we first take the black-box PIA approach by [ZTO21] and shift the attack set to be independent from the original data

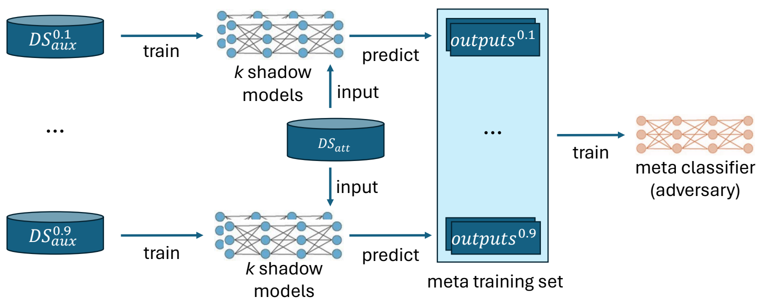


Figure 4.1: Black-box PIA in three steps: Train shadow models, generate output and train adversary.

distribution to further reduce the burden of pre-requisites on black-box PIAs. We further adopt the suggestion by [SE22] and take property inference as a logistic regression problem to directly infer specific property ratios through our meta-classifier, where others only consider a definite set of distributions and formulate the adversarial task as a classification problem. For defending against black-box PIAs, we adapt an adversarial learning strategy introduced by [GRLD20], where they directly include an adversarial loss term during model training to influence its memorization regarding specific properties. In their work they utilize this training framework to achieve fairness in decision tree models. Transitioning this process to PIA defense, we use a PIA adversary during training to punish a model, when it memorizes a distribution too much, leading to an increased adversary loss term. Using this process, we can dictate a model to learn a specific distribution output, and to hide its true distribution, regarding a sensitive property. Our goal in defending is comparable to the proposition by [ZCSZ22] but is applicable without actively removing training samples and we instead just reduce their influence to steer the property exposure.

4.3 Regression Property Inference Attacks

Given a trained target model, the overall aim of a PIA is to recover sensitive information about the model's training data. In contrast to other attacks, PIAs are not directed at properties of an individual data sample but rather at *global* properties of the training dataset. Examples for such critical properties include the gender ratio (male to female) in a dataset or the status of security patches in log files used by an intrusion detection system [GWY+18]. While some properties are binary, this work focuses on the more difficult problem of extracting continuous properties. Hence, we introduce PIAs as a regression problem, as proposed in [SE22].

4.3.1 Black-box attacker model

In this work, we focus on a black-box attacker model, in which an attacker does not have access to the target model itself, but can choose input values and observe the target model's output. This is a realistic scenario for many applications in which machine learning models are deployed as a service. In real-world applications, the internal model weights are often hidden from clients and only API-access is granted, meaning that requests are forwarded to the model internally and clients only receive its output. This helps the model owner to stay in control, and attacks such as model stealing [TZJ+16] can be prevented more easily.

The attacker has some information about the target model's training data or can access parts of it. This is an assumption also made for previous defense strategies [NSH18, SM21, TMS+21, SWDF23]. Otherwise, information about the training data can also be reconstructed with separate attacks [SSSS17], which is just as effective [LWH+22].

4.3.2 Attack Description

For our black-box PIA execution, we adapt the techniques from [ZTO21] for our scenario. The attack can be split in four steps, as described below and depicted in Figure 4.1.

Step 1: Training shadow models

Since the ultimate goal of an adversary $\mathscr{A}$ is to predict a property for a target model, $\mathscr{A}$ trains on the output of models with known property values – the shadow models. Hence, the first step in a PIA is to train multiple shadow models for different property values, i.e., shadow models are trained on training datasets with the respective property values. Since the adversary $\mathscr{A}$ is modeled as a regressor, we first create auxiliary training datasets with the property values $x \in [0.1, 0.2, \dots, 0.8, 0.9]$, i.e., each auxiliary training dataset DS^x_{aux} features ratio x regarding a predefined property. As an example, the property might be defined over the ratio of men to women in a dataset. To create an auxiliary dataset DS^x_{aux} in our experiments, we use the original training dataset and delete records until property value x is reached. This process is repeated for all $x \in [0.1, 0.2, \dots, 0.8, 0.9]$, such that the resulting 9 auxiliary training datasets DS^x_{aux} can be used to train shadow models. On each DS^x_{aux}, k shadow models are trained. For the datasets we have used in our experiments, $k = 200$ has proven useful. While a higher number for k might increase the utility of the advesary, it comes with higher computational costs.

By basing the auxiliary datasets on the original training datasets, we create worst-case adversaries, i.e., the strongest possible adversaries: Since the distribution of the DS^x_{aux} is as close as possible to the original training data distributions, the adversary learns the behaviour of very similar models to the attacked model. This is especially helpful when model owners defend their models during training. Since original training data must be available to the model owner during the training process, this prerequisite is easily fulfilled.

Step 2: Generating output from shadow models

The focus of this work is on a black-box attacker model, meaning that the adversary $\mathscr{A}$ does not have access to a target model itself but can only survey its output on a chosen input. Therefore, after training the 9*200 shadow models, they are each queried with the same input. This input can be arbitrary, as long as it yields a meaningful output when used as input to the shadow models. Since this input is an essential part of the black-box attack, we call it attack dataset DS_{att}. The output of the shadow models, labeled with the according property values $[0.1, 0.2, \ldots, 0.8, 0.9]$, is then stored as the training dataset for the adversary $\mathscr{A}$. The same process of training and generating output on shadow models is repeated on equally sized and non-overlapping auxiliary *test* datasets. For creating a test dataset for the adversary, we use 50 shadow models per property value, i.e., in total there are 9*200 shadow models to generate the meta training set and 9*50 shadow models to generate the meta test set.

Step 3: Training adversary on shadow model output

The adversary $\mathscr{A}$, or *meta-classifier*, is modeled as a simple ANN. The training dataset of the adversary (*meta training set*) consists of the outputs of the trained shadow models on the chosen input data, paired with the property values of their respective training datasets. The hyperparameters of the adversary such as neurons per layer and batch size are optimized via the framework keras-tuner[1]. The optimizer Adam is used to fit the model and early stopping is applied to avoid overfitting. After approximately 65 epochs, the meta-classifier $\mathscr{A}$ reaches its peak test R^2 value and the training stops.

Step 4: Attacking target model

After training the adversary $\mathscr{A}$, a target model m can be attacked by querying it with the attack dataset DS_{att}. The output of m then serves as the input for the adversary to infer the sensitive property, i.e., $\mathscr{A}(m(DS_{\text{att}})) = x$, where x is the value of the sensitive property inferred by the adversary $\mathscr{A}$.

1. https://keras.io/keras_tuner/,accessedonMarch30,2024

4.3.3 White-Box Benchmark PIA

In order to benchmark our novel black-box regression adversaries described above, we train white-box advesaries on the same datasets for comparison. In principle, we follow the work of [GWY+18], which is also based on shadow models. Hence, step 1 in Section 4.3.2 is identical in the white-box setting. Step 2 (generating output from shadow models) is not necessary, since the adversary is trained on the shadow model weights and biases itself – constituting the final step of training a white-box adversary. The white-box attack is carried out by directly using the weights and biases of a target model as input to the trained adversary.

4.4 Defending Black-Box PIA

To defend against black-box PIAs, we propose a form of adversarial training. We borrow the technique of [GRLD20], originally intended to make models *fair*, in the sense of making decisions independent from pre-specified sensitive characteristics in the data. Grari et al. design an adversary with the task of deducing the sensitive data property from the model output for Gradient Boosted Trees (GBTs). Using this adversary during training, they create additional gradients for the trained model – to minimize the success of the adversary, i.e., to minimize the influence of the sensitive property on the model's output. A parameter λ is introduced which controls the tradeoff between the two competing training goals model performance and property suppression.

We have extended the strategy of [GRLD20] for GBTs to defend ANNs against PIAs. For defending a target ANN during training, we can simply modify its loss function. To be precise, we define two terms within the loss function: A first term punishing the model if its prediction deviates from the labels of the training data, and a second term punishing the model if the PIA adversary infers another value than the predefined target distribution from the model's output. As in the work of [GRLD20], the two terms are weighted with $(\lambda - 1)$ and λ, allowing to find an optimal tradeoff between model optimization and PIA defense.

When computing the loss ℓ as a mean squared error (mse), this yields

$$\ell = (1-\lambda) * mse(y_{true}, y_{pred}) + \lambda * mse(adv_{tar}, adv_{pred})$$

for training data labels y_{true} and model predictions y_{pred}, the adversary's target distribution adv_{tar} and the adversary's current prediction adv_{pred} based on the model's output.

4.5 Experiments

We perform our experiments on the three datasets Adult, UTKFace and CIFAR-10. We have performed our experiments on linux machines using Python 3.10 and the latest versions of TensorFlow and keras[2]. The defense experiments have been performed on 10 target models per property value and dataset. The target models are trained on a different portion of the datasets than the shadow models (i.e., the auxiliary test datasets as described in Section 4.3.2, Step 2) to avoid side effects.

4.5.1 Datasets

Adult is a dataset consisting of 32,561 records from the US census [K+96]. The 14 attributes include information about the individual gender, work hours per week and education level. The machine learning task is to predict whether an individual earns more than 50k dollars per year. For this tabular dataset, we use random forest classifiers, reaching an accuracy of 85%.

As the sensitive attribute, we choose the attribute *gender* in the adult dataset, i.e., the adversary's goal is to predict the ratio of male:female records in the training dataset. For the *attack dataset* which the adversary uses to generate output from target models, we generate a synthetic dataset with 10,000 records based on Adult using a conditional generative adversarial network (CTGAN) [XSCIV19].

UTKFace consists of 20,000 color images with annotations of age, gender and ethnicity [ZSQ17]. We choose the ML task of inferring the correct gender from an image. ANNs with convolutional layers are used for recognizing the images, reaching a test accuracy of 90%.

We define our sensitive property over the *age* attribute, i.e., the property is defined by the distribution of old to young instances in the data. For simplicity, we consider all *age* labels above 59 as old. Our attack dataset is based on the *Labeled faces in the wild* dataset [HMBLM08], containing 13,233 images of 5,749 different people.

CIFAR-10 contains 60,000 color images for image recognition tasks with 32x32 pixels each [Ale09]. The dataset is grouped into 10 classes, such as *airplane*, *horse* and *truck*. For the target and shadow models, we use ANNs with the same architecture as for UTKFace above, reaching accuracies of 70% (random guessing would amount to 10% accuracy).

The sensitive attribute we define in this dataset is the amount of animals in the training dataset, i.e., the adversary's task is to predict the ratio of animal:non-animal images in

2. The code is available open source at [Sto25b]

Access	white-box	black-box	
Defense	none	none	$\lambda = 0.15$
Adult	–	0.72	–
UTKFace	0.86	0.63	0.070
CIFAR-10	0.60	0.64	0.069

Table 4.2: Adversary performance measured as R^2 on test data, comparison black-box and white-box. Both the white-box attack and the black-box defense are not applicable to the tree-based models for the Adult dataset.

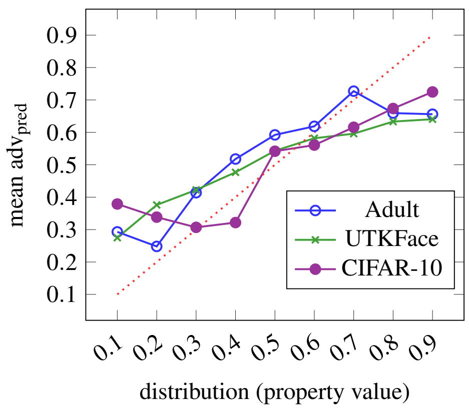
the training dataset of the target model. The attack dataset which the adversary uses to generate output from the target model is based on the CIFAR-100 dataset [Ale09]. We randomly select 5,040 samples of 21 CIFAR-100 classes, which share similarities with CIFAR-10 classes, e.g., images of the CIFAR-100 class *lion* have similar features as the CIFAR-10 class *cat*.

4.5.2 Regression PIA results

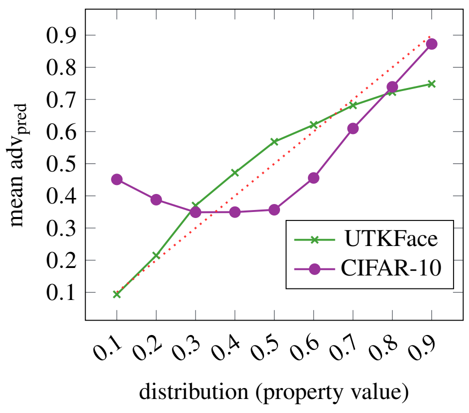
The results of our attack experiments are summarized in the first two columns of Table 4.2. Following the regression approach explained in Section 4.3, we have trained PIA adversaries both in white-box and black-box scenarios[3]. The black-box adversaries reach R^2 values from 0.63 (UTKFace) to 0.72 (Adult). Their performance is visualized in Figure 4.2a: An ideal adversary would infer all property values correctly, as shown by the dotted red line. It is apparent that all three adversaries perform best in the mid-ranges of property values (0.2–0.7), where their predictions are closest to the ideal. For CIFAR-10, the largest deviation is at property value 0.1, where the mean prediction amounts to 0.38. The smallest deviations are observed for property value 0.6, where all three adversaries' mean predictions deviate less than 5% from the correct value.

Focusing on the first column of Table 4.2, we can see that the white-box adversary for UTKFace outperforms its black-box counterpart (R^2 of 0.86 compared to 0.63), while the performance of the white-box adversary for CIFAR-10 is worse than the black-box version (R^2 of 0.60 opposed to 0.64). The great performance of the white-box UTKFace adversary is also reflected in Figure 4.2b, where the plotted deviations from the ideal predictions (dotted red line) are small or even non existent in the case of property value 0.1. At the same time, the CIFAR-10 white-box adversary performs worst at property value 0.1, its mean prediction deviating even more from the truth than its black-box counterpart in

3. The white-box approach was not applicable to the GBTs of the adult dataset, since the proposal in [GWY+18] is designed for ANNs.



(a) Black-box.



(b) White-box.

Figure 4.2: Performance of adversaries, mean adversary output on test data for different distributions (resp. property values) for the two attack scenarios.

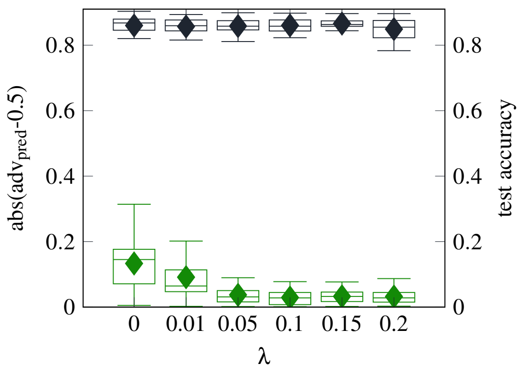
Figure 4.2a. Interestingly, the CIFAR-10 white-box adversary obtains its best predictions for property value 0.9, contrary to the UTKFace white-box adversary.

4.5.3 Regression black-box PIA defense results

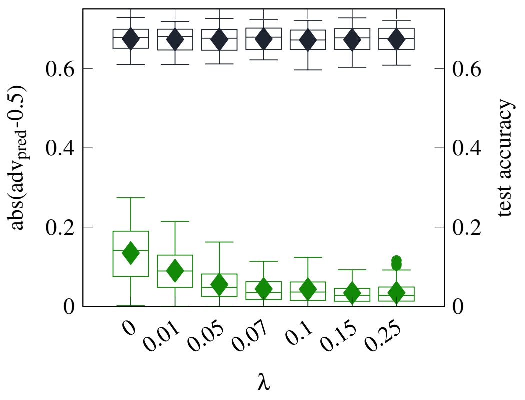
We have implemented the defense strategy of Section 4.4 and conducted experiments for different values of λ. To recapitulate, the higher the value of λ, the more the trained model is defended against a PIA adversary, hence $\lambda = 0$ implies a regular training without any defense mechanisms. All experiments were run with the target property value 0.5.

To show the effect of our adversarial training, we plot the adversarial outputs after defending target models with $\lambda = 0.15$ in Figure 4.3: The plot is very different from the original adversary performance in Figure 4.2a, with mean adversary outputs close to the target 0.5 for target models with all property values. This underlines the results in the third column of Table 4.2 with an adversary R^2 of 0.07 on defended models for both datasets UTKFace and CIFAR-10.

More detailed results for different values of λ are presented in Figure 4.3a for UTKFace and Figure 4.3b for the CIFAR-10 dataset. The green boxplots at the bottom of both figures represent the distance of adversary outputs to the target value 0.5 – as λ increases along the x-axis, this distance decreases. The black boxplots at the top of the figures show the accuracies of the defended models. For both datasets, we can observe practically no accuracy decrease: The mean accuracy even increases for the UTKFace models from 86.19% ($\lambda = 0$) to 86.60% ($\lambda = 0.15$), before it decreases slightly to 84.91 for $\lambda = 0.2$. For CIFAR-10, the accuracy decreases slightly from 67.46% ($\lambda = 0$) to 67.33% ($\lambda = 0.25$).



(a) UTKFace.



(b) CIFAR-10.

Figure 4.3: Accuracy of defended models for different values of lambda (black) and absolute divergence from target 0.5 of the adversary for defended models (green) for two data sets.

4.6 Discussion

Regarding the attack success rates, the black-box and white-box adversaries are in the same range of 0.6 to 0.86 R^2. While the *age* property for the UTKFace dataset is easily extracted in the black-box scenario (highest R^2: 0.86), the *animal* property of CIFAR-10 seems to be harder to extract for an adversary, especially for property values below 0.3 (both in the black-box and white-box scenarios). The reasons behind this are guesswork; one hypothesis could be that as long as the share of animal images during CIFAR-10 training is below a certain threshold, the ANN does not account for typical animal features to an extend which is inferrable for an adversary.

In general, the attack success rates are hard to compare to related work, since PIAs are only considered as a classification problem, not covering the more natural continuous value range of property distributions. To the best of our knowledge, [ZTO21] is the only work with a similar PIA experiment. In reference to their “fine-grained” attack using 5 classes for the property *gender* in the Adult dataset, we calculate approximate accuracy values for our attackers by classifying our adversary outputs into 4 classes across the range 0–1. The value from [ZTO21] is the average value of the rightmost column in Table 7, i.e., the average accuracy across the 5 classes. The results (see Table 4.3) exhibit similar accuracy rates across all adversaries (39.8%–67.6%), although Zhang et al.’s work seems to outperform our black-box attacks. The two white-box attacks show significantly higher accuracy rates than the black-box attacks. However, we need to stress that the accuracy values are extracted from our regression outputs and that our adversaries have not been optimized to achieve high accuracies in a classification task.

	black-box		white-box
	related work	ours	ours
Adult	50.6%	39.8%	–
UTKFace	–	44.9%	67.6%
CIFAR-10	–	49.2%	62.4%

Table 4.3: Approximated accuracies for our attacks to establish comparability to related work. The value for related work is extracted from [ZTO21], Table 7.

In our defense experiments, we have shown that the adversarial approach, actively suppressing property information in target model outputs, works well and does not negatively affect the performance of defended target models. Across both datasets, $\lambda = 0.15$ has proven to create a reasonable tradeoff during training, minimizing the adversary's R^2 to 0.07 while harming the target models' performance by less than 0.2 percentage points on average. As Figure 4.3a exhibits, using a higher λ than 0.15 is not necessary, since the difference in adversary performance is negligible (R^2 of 0.067 for $\lambda = 0.25$ instead of R^2 of 0.069 for $\lambda = 0.15$) and two outliers imply possibly unwanted behavior for higher values of λ. Another notable observation from Figure 4.3a and Figure 4.3a is that not only the accuracy values are stable, but also their deviations do not increase for bigger values of λ.

For demonstration purposes, we have used 0.5 as the target property value across all defense experiments. In practice, this value could be either randomly chosen from the range 0–1 for each target model individually, or set to some other constant value. Depending on the use case, one option might make more sense than the other.

In [SWDF23], the authors have demonstrated how it does not suffice to harden a target model against a single white-box adversary to defend a whole class of PIA adversaries. For black-box adversaries, this is different, since the information available to the adversary is a lot more sparse than in white-box scenarios (model output vs. all trained weights and biases). Therefore, adversaries cannot circumvent the adversarial defense by focusing on another part of available information, as has been shown for the white-box case in Section 6 of [SWDF23]. We were able to confirm this experimentally by validating that defending a target model against one black-box adversary limits the capabilities of another adversary with the same task[4]. This shows that the defense mechanism in this work generalizes well, in contrast to the white-box defense presented in [SWDF23].

4. Details on this experiment are presented in Appendix B.

4.6.1 Future Work

Through our experiments, we have demonstrated the feasibility of regression PIAs. Although this will not prevent attackers from executing them, the process of implementing the attack takes a lot of effort, entailing the identification of a target model's training data distribution, creating shadow models, fine-tuning the architecture of the adversary model, etc. Follow-up work could investigate whether this effort could be limited, while maintaining the success rates shown in this work. Inspiration could be taken from [LZ21], where a "boundary attack" for membership inference is presented, which bypasses the creation and usage of shadow models altogether.

Also, our defense mechanism could be transferred to a hybrid scenario. Instead of using a static adversary during the adversarial training of a target model, the adversary is retrained on the modified output of the target model in every epoch of the target model training, as in [GRLD20]. Such a hybrid adversarial training could potentially further reduce the leaked property information of a target model, although further investigation is necessary.

Last but not least, the adversarial training for fairness in [GRLD20] could have side effects on the success rate of PIAs. Since Grari et al. train their models to yield outputs independent from a sensitive property p, it would be interesting to investigate whether their approach could also defend a target model against a PIA focussing on property p.

4.7 Conclusion

In this work, we have expanded upon existing black-box PIAs by using an arbitrary attack dataset, which can be based on other datasets than the training dataset. As the natural fit for many ratio-based properties, we have modeled the PIAs in this work as regression problems. We have explored a defense mechanism based on adversarial training which hardens a target model against black-box PIAs during its training process. We have evaluated our approach on three datasets, comparing the attack against white-box benchmarks and related work. In our experiments, we have shown our defense scheme to be both effective (by decreasing the adversary's performance from an R^2 of 0.63–0.64 to 0.07) and practical, decreasing the mean accuracy of target models by less than 0.2 percentage points.

Acknowledgements

The authors would like to thank Jona Zantz for his helpful comments and insights.

References

[Ale09] Krizhevsky Alex. Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*, 2009.

[AMS$^+$15] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[ARC19] Mohammad Al-Rubaie and J Morris Chang. Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, 17(2):49–58, 2019.

[GRLD20] Vincent Grari, Boris Ruf, Sylvain Lamprier, and Marcin Detyniecki. Achieving fairness with decision trees: An adversarial approach. *Data Science and Engineering*, 5(2):99–110, 2020.

[GWY$^+$18] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[HMBLM08] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[HMP$^+$23] Valentin Hartmann, Léo Meynent, Maxime Peyrard, Dimitrios Dimitriadis, Shruti Tople, and Robert West. Distribution inference risks: Identifying and mitigating sources of leakage. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 136–149. IEEE, 2023.

[K$^+$96] Ron Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

[LWH$^+$22] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. ML-Doctor: Holistic risk assessment of inference attacks against machine learning models. In *USENIX Security*, 2022.

[LZ21] Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 880–895, 2021.

[NSH18] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *CCS*, 2018.

[SE22] Anshuman Suri and David Evans. Formalizing and estimating distribution inference risks. *Proceedings on Privacy Enhancing Technologies*, 2022.

[SM21] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *USENIX Security*, 2021.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[Sto25] Joshua Stock. Regression property inference experiments. https://github.com/joshua-stock/regression-property-inference, 2025.

[SWDF23] Joshua Stock, Jens Wettlaufer, Daniel Demmler, and Hannes Federrath. Lessons learned: Defending against property inference attacks. In Sabrina De Capitani di Vimercati and Pierangela Samarati, editors, *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT)*, pages 312–323. SCITEPRESS, 2023.

[TMS+21] Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. Mitigating membership inference attacks by self-distillation through a novel ensemble architecture. *USENIX Sec.*, 2021.

[TZJ+16] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[XBJ21] Runhua Xu, Nathalie Baracaldo, and James Joshi. Privacy-preserving machine learning: Methods, challenges and directions. *arXiv preprint arXiv:2108.04417*, 2021.

[XSCIV19] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In *Advances in Neural Information Processing Systems*, 2019.

[ZCSZ22] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. Property inference attacks against gans. In *30th Network and Distributed System Security Symposium (NDSS 2022)*, 2022.

[ZSQ17] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818, 2017.

[ZTO21] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of dataset properties in {Multi-Party} machine learning. In *30th USENIX security symposium (USENIX Security 21)*, pages 2687–2704, 2021.

5 The Applicability of Federated Learning to Official Statistics

Citation	Joshua Stock, Oliver Hauke, Julius Weißmann, and Hannes Federrath (2023). The Applicability of Federated Learning to Official Statistics. In *Intelligent Data Engineering and Automated Learning (IDEAL)*. Lecture Notes in Computer Science, vol 14404. Springer, Cham, ISBN: 978-3-031-48232-8, pages 70–81. DOI: 10.1007/978-3-031-48232-8_8
Venue	24th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'23)
Ranking	**C** (CORE 2023), **C** (CORE 2021), **B2** (Qualis 2016), **C** (ERA 2010)
Status	Published. Reproduced with permission from Springer Nature. This article is licensed by Springer Nature Customer Service Center GmbH, and is NOT part of the overriding OA/Creative Commons license.
Publication Type	Research Paper
Aim	The aim of this paper was to analyze the applicability of federated learning to official statistics. More specifically, its benefits and practical limitations should be explored. This analysis was framed in a case study with three different use cases, two of them with a focus on comparing the performance of centralized machine learning to federated learning. The third use case was aimed at showcasing practical limits when implementing a federated learning process.
Methodology	For each use case, the case study provides experiments comparing the performance of ML models trained in the conventional centralized way and a federated learning simulation (in which the data sets were split and distributed among the simulated clients). While the first two use cases were entirely simulated, the third use case was set in a more realistic scenario where the direct access to training data was not possible during development.

Continued on next page

Contribution	The three use cases are based on a medical insurance data set, a fine dust pollution data set and a mobile radio coverage data set. These domains are all close to official statistics. In each use case, the performance of federated learning was close to the benchmark of centralized learning techniques, indicating federated learning as a useful enabling technology, especially if training data could not be transferred due to data protection rules. The third simulation of the case study allowed to draw conclusions regarding the applicability of federated learning and potential practical limitations, such as the difficulty of hyperparameter optimization without any training data at hand. The code for the simulations is available open source [Sto23].
Co-authors' contribution	This article was co-authored by Oliver Hauke and Julius Weißmann who provided their knowledge about ML and national statistical offices. Both assisted in writing the paper, especially Section 5.5 (Implications for Official Statistics). Hannes Federrath provided editorial guidance.

Abstract

This work investigates the potential of federated learning (FL) for official statistics and shows how well the performance of FL models can keep up with centralized learning methods. At the same time, its utilization can safeguard the privacy of data holders, thus facilitating access to a broader range of data and ultimately enhancing official statistics. By simulating three different use cases, important insights on the applicability of the technology are gained. The use cases are based on a medical insurance data set, a fine dust pollution data set and a mobile radio coverage data set – all of which are from domains close to official statistics. We provide a detailed analysis of the results, including a comparison of centralized and FL algorithm performances for each simulation. Our key observations and their implications for transferring the simulations into practice are summarized. We arrive at the conclusion that FL has the potential to emerge as a pivotal technology in future use cases of official statistics.

5.1 Introduction

The aim of national statistical offices (NSOs) is to develop, produce and disseminate high-quality official statistics that can be considered a reliable portrayal of reality [YTB⁺22]. In order to effectively capture our rapidly changing world, NSOs are currently undergoing a process of modernization, leveraging new data sources, methodologies and technologies.

NSOs have effectively extracted information from new data sources, such as scanner data[1] or mobile network provider (MNO) data[2]. However, the potential of numerous other data sources, including privately held data[3] or data from official entities, remains largely untapped. Legal frameworks, which are fundamental to official statistics, only adapt slowly to changing data needs and currently hinder access to valuable new data sources. Cooperation with potential data donors faced restrictions due to concerns about disclosing individual business interests, privacy or confidentiality.

In contrast, the methodology employed by NSOs is evolving rapidly, with machine learning (ML) gaining substantial popularity and, as a result, undergoing a process of establishment.

1. Scanner data in consumer price statistics and for determining regional price differences https://www.destatis.de/EN/Service/EXSTAT/Datensaetze/scanner-data.html, accessed on July 17, 2023
2. Use of MNO data https://cros-legacy.ec.europa.eu/content/12-use-mno-data_en, accessed on July 17, 2023
3. Guidance on private sector data sharing https://digital-strategy.ec.europa.eu/en/policies/private-sector-data-sharing, accessed on July 17, 2023

ML has been applied in various areas of official statistics (e.g. [DB17, BDF18]), and new frameworks such as [YTB^{+}22] address the need to measure the quality of ML.

Within official statistics, ML tools have proven effective in processing new data sources, such as text and images, or enabling the automation of statistical production tasks, including classifying information or predicting not (yet) available data.

Federated learning (FL) is an emerging approach within ML that provides immense unexplored potential for official statistics. It addresses the challenge of extracting and exchanging valuable global information from new data sources without compromising the privacy of individual data owners. Introduced in [MMR^{+}17], FL enables collaborative model training across distributed data sources while preserving data privacy by keeping the data localized. In scenarios where external partners are unwilling to share individual-level information due to regulatory or strategic considerations, but still aim to analyze or disseminate global insights in their field of application, NSOs can offer trustworthy solutions utilizing FL. In return, FL empowers contributing NSOs to integrate new data sources in statistical production.

This work investigates the methodological and practical potential of FL for NSOs to leverage new data sources. The primary contribution of this work involves a comprehensive investigation through three simulations that address current data needs representative for official statistics. In the first simulation related to **health**, individual healthcare cost are predicted utilizing a regression. In the second simulation related to **sustainability**, the newest fine dust pollution levels are classified based on meteorological data. In the third simulation related to **mobility**, the daily range of movement of mobile phone users are classified by MNO data. The first two simulations focus on assessing the estimation performance achieved by FL in comparison to centralized models that have complete access to all available data. The third application presents valuable insights and lessons learned from the implementation of FL, involving the active participation of a real external partner. We draw conclusions on the applicability of FL in NSOs in Section 5.5, which are summarized in Section 5.6.

5.2 Background

Before presenting the simulated use cases in Section 5.3, this section provides an overview of FL and privacy challenges with ML.

5.2.1 Federated Learning

In FL, a centralized server (or aggregator, in our case a NSO) coordinates a process of training a ML model (mainly deep ANNs) by initializing a global model and forwarding it to the data owners (clients). In each training round, each client trains the model with their private data and sends the resulting model back to the central server. The central server uses a FL method to aggregate the updates of the participants into the next iteration of the global model and starts the next round by distributing the updated model to the clients. This process is repeated to improve the performance of the model.

NSOs primarily strive to generate global models that accurately represent the available data, which, in our setting, is distributed among multiple clients. Thus, we compare the performance of FL to models with access to the combined data of all clients. Alternatively, if upcoming applications seek to supply each client with the an optimized individual model by leveraging information from the other clients, *personalized* FL can be used. It is not covered in this paper but can be found in [KKP20, HGLM18].

5.2.2 Privacy Challenges with Machine Learning

When training data for an ML model is distributed among multiple parties, the data needs to be aggregated on a central server prior to applying traditional ML algorithms. FL has become a popular alternative to this approach, as it allows to train a model in a distributed way from the start, without the need to aggregate training data first. Thus, using FL has the privacy advantage that there is no need to exchange private training data. Instead, data holders can train a global model collaboratively in a distributed fashion, without giving any party access to their individual data set.

But although FL makes sharing private training obsolete, there are other privacy challenges inherent to ML which have also been observed for FL. While ML models are always trained to fulfill a dedicated task, often more information than strictly necessary for fulfilling the task is extracted into the model weights during training [SRS17]. This excessive, and potentially private, information in the model weights is called privacy leakage.

Examples on how this privacy leakage can be leveraged are attacks such as *training data extraction* [ZLH19], which allows to extract actual training data records from a trained model. Another known attack is *model inversion* [HAPC17], where repeated requests to the model are used to reconstruct class representatives. This is especially feasible if the records are very similar to each other, as shown in [SSSS17]. The authors of [SSSS17] have also introduced the *membership inference* attack, which has evolved to one of the most important ML privacy attacks. This is because membership inference is a simple attack aiming at individual training data records: the attack's target is to decide whether a specific data record was part of the training data. Due to its simplicity, it is often used as a benchmark for general privacy leakage. Building on the original proposal, other works have transferred membership inference attacks to the FL scenario [NSH19]. Last but not least, *property inference* attacks [MSDCS19] allow to deduce statistical properties of the target model's training data. This is especially relevant in FL scenarios, where the characteristics of each client's local data set can be highly sensitive, e.g., in medical domains.

The applicability of these attacks depends on the use case, the type of model and other factors. Concerning attacker models, i.e., the scenario in which an attack is executed, some FL-specific attacks rely on a malicious aggregator. Nonetheless, all attacks mentioned above also work in an environment where not the aggregator, but one of the FL clients is the attacker. Hence, even if the aggregator can be trusted, e.g., because the aggregator's role is assumed by a NSO, these attacks can still be executed by other FL clients. Analyzing the individual privacy leakage of the simulated use cases in this paper are out of scope, nonetheless we want to stress the importance of awareness towards these privacy issues, which persist in many ML (including FL) scenarios. In an FL process, the least one can do is communicating potential privacy risks to clients. Beyond this, there are some strategies under the umbrella term *privacy-preserving machine learning (PPML)* which might help in mitigating these attacks, although most of them are not yet computationally feasible enough for practical applications [YZH21].

5.2.3 Frameworks

In our simulations, we use the frameworks TensorFlow[4] for ANNs and TensorFlow Federated[5] for FL. For data processing, we use scikit-learn[6] and PyCaret[7] for automizing benchmark experiments in the centralized settings.

The code we have written for this work is openly available on GitHub [Sto23][8].

4. TensorFlow https://www.tensorflow.org/, accessed on July 17, 2023
5. TensorFlow Federated: Machine learning on decentralized data https://www.tensorflow.org/federated, accessed on July 17, 2023
6. scikit-learn, Machine Learning in Python https://scikit-learn.org/, accessed on July 17, 2023
7. Pycaret https://pycaret.org/, accessed on July 17, 2023
8. Note that for the mobile radio coverage simulation, the code has only been executed locally on the private data set, hence it is not included in the repository.

5.3 Simulations

Most relevant for NSOs is *cross-silo* FL, where a few reliable clients train a model, e.g. official authorities. In contrast, *cross-device* FL uses numerous clients, e.g. smartphones, to train a model. To analyze the potential of cross-silo FL for official statistics, we run simulations with three different data sets. For each use case, we first compute benchmarks by evaluating centralized ML models, i.e., models which are trained on the whole data set. Afterwards, we split the data set and assign the parts to (simulated) FL clients for the FL simulation. This way, we have a basis for interpreting the performance of the model resulting from the FL training simulation. The performance metrics of the trained ML models (including coefficient of determination R^2 or accuracy) are computed on test sets of each data set.

5.3.1 Medical insurance data

The demand for timely and reliable information on public health is steadily increasing. The COVID-19 pandemic has significantly accelerated this trend, bringing questions about the financial feasibility of our healthcare system and the availability of medical supplies to the forefront.

Thus, our first experiment focuses on modeling a regression problem related to healthcare by considering the following question: Given individuals' health status characteristics, what is the magnitude of their insurance *charges*? Our study aims to address two primary questions. Firstly, we explore the suitability of ANNs in comparison to other models for the regression task. Secondly, we assess the feasibility of utilizing a simulated decentralized data set with an FL setting to tackle the problem.

Data set The first data set links medical insurance premium (*charges*) to related individual attributes[9]. Considered are the six features *age*, *sex*, *bmi* (body mass index), *children* (count), *smoker* (yes/no) and four *regions*. In our studies, the feature *region* was excluded during training and solely utilized for partitioning the data within the FL setting. It consists of 1338 complete records, i.e. there are no missing or undefined values. Also, the data set is highly balanced: The values in *age* are evenly dispersed, just as the distribution of male and female records is about 50/50 (attribute *gender*) and each *region* is represented nearly equally often. The origin of the data is unknown, however its homogeneity and integrity suggest that it has been created artificially.

9. US health insurance dataset https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset, accessed on July 17, 2023

Data preprocessing We encode the binary attributes *sex* and *smoker* into a numeric form (0 or 1). The attributes *age*, *bmi* and *children* are scaled to a range from 0 to 1. In the centralized benchmarks, we use the *region* attribute one-hot-encoded.

Setup In this research, we aim to investigate the suitability of ANNs for estimating insurance *charges* and explore the extent to which this problem can be addressed using a FL approach. To achieve this, we compare different models and evaluate their performance.

In this study, a basic fully connected ANN architecture, that takes five input features, was utilized. The network consists of three hidden layers with 40, 40, and 20 units in each respective layer. Following each layer, a Rectified Linear Unit (ReLU) activation function is applied. The final output layer comprises a single neuron. To optimize the network, the Adam optimizer with a learning rate of 5e-2 was employed. In the federated task, we utilized the same initial model but integrated FedAdam for server updates. This decision was based on previous research [RCZ+20], which emphasized the benefits of adaptive server optimization techniques for achieving improved convergence.

In the centralized approach, we allocate a training budget of 100 epochs. In contrast, the federated approach incorporates 50 rounds of communication between the client and server during training. Each round involves clients individually training the model for 50 epochs. To evaluate our results in both scenarios, we employ 5-fold cross-validation for testing. To track the running training, 10% evaluation data is used by each client in the FL setting and 20% is used in the centralized scenario. It is neglected in calculating the final test performance. The remaining shallow learning models underwent HPO using a random search approach with a budget of 100 iterations. Similar to the ANNs, we conducted evaluation using 5-fold cross-validation.

Results We conducted a performance comparison of the models based on their 5-fold cross-validation R^2 scores and considered their standard deviation (see Table 5.2). The Random Forest model achieved the highest performance with R^2 of 0.845, closely followed by XGBoost and Decision Tree, which scored 0.2 and 0.5 percentage points lower, respectively.

The ANN model achieved an R^2 of 0.815, indicating a performance 3.5 % worse than the best model. However, it still provides a reasonable result compared to K-Nearest Neighbors (KNN) and Linear Regression, which obtained significantly lower R^2 scores of 12 % and 13.8 %, respectively.

The federated ANN demonstrated an R^2 of 0.784, slightly lower than the centralized ANN but 7.2 % worse than the Random Forest model. Notably, the Federated ANN exhibited

Model	**R^2(± std)**	**Rel. loss (%)**
ANN	**0.815 (0.04)**	3.5
ANN (federated)	**0.784 (0.03)**	7.2
random forest	**0.845 (0.05)**	0.0
XGBoost	0.843 (0.04)	0.2
decision tree	0.841 (0.04)	0.5
k-nearest neighbors	0.744 (0.06)	12.0
linear regression	0.728 (0.06)	13.8

Table 5.2: Performance comparison of different prediction models for the medical insurance use case. The performance is quantified using R^2, along with the corresponding standard deviation (std). Additionally, the relative loss to the best centralized model (rel. loss) is reported.

a lower standard deviation of 0.04 compared to the centralized ANN (0.05) and also outperformed the Random Forest model (0.05) in this regard.

Discussion Based on the research questions, we can draw clear conclusions from the findings presented in Table 5.2. Initially, we compared the performance of different models, including a simple ANN. Although the random forest model outperformed others, its performance was only 3.5 % higher, distinguishing it significantly from models such as KNN and linear regression, which performed 12 % and 13.8 % worse than the random forest, respectively.

The observed performance decrease from 0.815 to 0.784 in the FL approach can be attributed to the training process and falls within a reasonable range. Considering the privacy advantages of FL, the 7.2 % accuracy loss compared to the best model is acceptable, particularly when taking into account the reduction in standard deviation from 0.05 to 0.04.

Although this example is hypothetical, it highlights the potential benefits and importance of FL in official statistics. It showcases how FL provides access to crucial data sets for ML while maintaining nearly negligible loss in accuracy compared to a centralized data set.

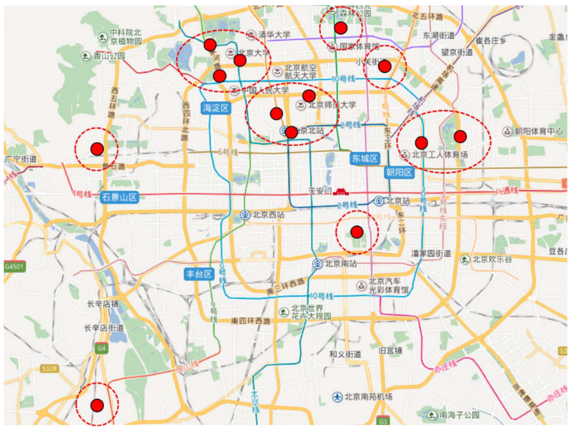


Figure 5.1: Location of meteorological stations for simulation 2 (fine dust pollution) on a map of Bejing, China. 12 of the 13 stations are included in the public data set which we have used for our simulations. The dashed lines mark *regions* of the “Region-Learning” approach in [HGLM18]. Image source: [HGLM18].

5.3.2 Fine dust pollution

Reducing air pollution is a significant part of the Sustainable Development Goals (SDGs) established by the United Nations[10]. To measure progress toward achieving SDGs, NGOs and other data producing organizations developed a set of 231 internationally comparable indicators, including *annual mean levels of fine particulate matter (e.g. $PM_{2.5}$ and $PM_{2.5}$)*. [HGLM18] showed that personalized FL can be used to extract timely, high frequent and more accurately than models using centralized data.

In our second use case, we additionally provide a comparison between centralized and FL models (without personalization) and make the developed code and methods accessible. It should be noted that we utilize a slightly different data set and methodology compared to [HGLM18]. We model a classification task in which the current fine dust pollution is inferred based on meteorological input data. More precisely, 48 consecutive hourly measurements are used to make a prediction for the current $PM_{2.5}$ pollution (the total weight of particles smaller than 2.5 μm in one m^3). The output of the predictor is one of the three classes *low*, *medium* or *high*. The threshold for each class are chosen in a way such that the samples of the whole data set are distributed evenly among the three classes.

10. Air quality and health https://www.who.int/teams/environment-climate-change-and-health/air-quality-and-health/policy-progress/sustainable-development-goals-air-pollution, accessed on July 17, 2023

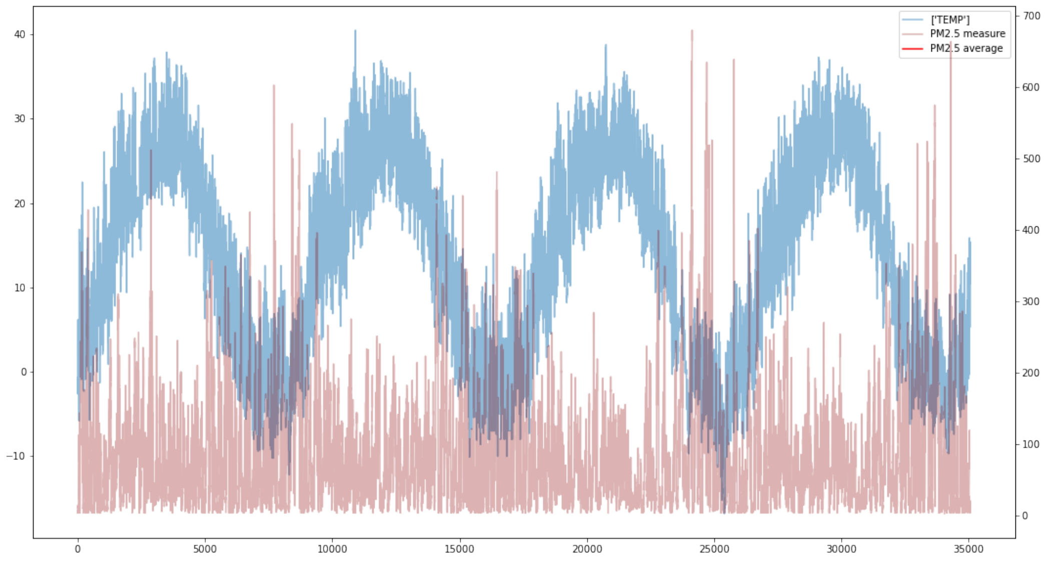


Figure 5.2: Example plot for the data of one meteorological station and the two features $PM_{2.5}$ and temperature. The four year time span is clearly visible by the temperature wave, due to hot summers and cold winters.

Data set The data set we use is a multi-feature air quality and weather data set [ZGD+17] which is publicly available online[11]. It consists of hourly measurements of 12 meteorological stations in Beijing, recorded over a time span of 4 years (2013–2017). Figure 5.1 depicts the locations of the 12 stations in Beijing. In total, more than 420 000 data records are included in the data set. Although some attributes are missing for some data records, most records have data for all of the 17 attributes. An example plot for the two attributes $PM_{2.5}$ and temperature is shown in Figure 5.2.

Data preprocessing To complete the missing data records, we use linear interpolation. We use one-hot encoding for the wind direction attribute. All other features are scaled using a standard scaler implementation. For the attributes PM_{10}, SO_2, NO_2, CO and O_3, we observe a high correlation with the target attribute and thus exclude them from training. 80% of the data are used as training data, the rest is used as test data.

Setup As in the first use case, we implement a centralized learning benchmark and compare it with a FL approach. We model one FL client per meteorological station and split the data accordingly, while the benchmark model is trained with data from all 12 stations. In both settings, we use an ANNs with long-short term memory (LSTM) layers

11. Beijing multi-site air-quality data set https://www.kaggle.com/datasets/sid321axn/beijing-multisite-airquality-data-set, accessed on July 17, 2023

Model	**Accuracy (± std)**	**Precision (± std)**	**Recall (± std)**	**Rel. loss (%)**
ANN	**72.4**% (4.92)	**72.8%** (8.66)	**72.3%** (8.10)	0.0
ANN (fed.)	**68.0%** (2.38)	**67.9%** (10.05)	**68.0%** (9.59)	5.9-6.7

Table 5.3: Performance in the fine dust pollution simulation. The span of the relative loss refers to all three metrics.

true class / **predicted class**	**low**	**medium**	**high**
low	114450	23712	5769
medium	23635	80718	33754
high	1944	24629	111581

Table 5.4: Exemplary confusion matrix for one of the five models in the cross-validation training of the centralized model for the fine dust pollution use case.

and apply 5-fold cross validation. The architecture of the ANNs is similar across both settings and has been manually tuned to reach a good performance. For the same reasons as in the first use case, we use the Adam optimizer and apply a learning rate of 0.05 on the server and 0.005 on the client. The client learning rate is decreased every 64 epochs by a factor of 10 to facilitate fine tuning in later stages of the training. The total training budget we have allocated is 10 epochs for centralized learning and 200 epochs for FL.

Results A summary of our results for the fine dust pollution use case is provided in Table 5.3. Depicted are the means of our 5-fold cross validation experiments.

The centralized learning benchmark reaches a mean classification accuracy of 72.4%, with similarly high numbers for precision and recall (72.8%, respectively 72.3%). In comparison, the FL classifier reaches a performance of both an accuracy and a recall of 68.0% and a precision of 67.9%. The relative standard deviation is higher in the FL scenario for all three metrics, reaching from +2.67 percentage points (accuracy) to +2.9 percentage points (both precision and recall).

An exemplary confusion matrix for one of the five resulting models of the centralized learning is depicted in Table 5.4. Most misclassifications are made for the *medium* class. The same could be observed for the other models (both in centralized and FL).

Discussion Compared to the first use case, the training data base has been significantly larger. With 12 clients, there have also been 4 times as many participants in the FL scenario as in the first use case. Still, the performance decrease is small, with an accuracy of 68.0% (FL) compared to 72.4% in the centralized training scenario.

Apart from preprocessing the data set, another time consuming part of the engineering was tuning the hyperparameters of the FL training. Tools for automatic FL HPO were out of scope for this work, thus it was necessary to manually trigger different trial runs with varying hyperparameters.

Comparison with literature The authors of [HGLM18] compare the results of their personalized FL strategy “Region-Learning” to a centralized learning baseline and standard FL. Although according to the authors, their personalized FL approach outperforms the other two approaches (by 5 percentage points compared to standard FL), we want to stress that Region-Learning has another goal than standard FL – namely multiple specialized models, and not one global model as in standard FL and most use cases for official statistics (also see Section 5.2.1).

Furthermore, Hu et al. have not provided sufficient information to properly retrace their experiments. Especially the number of classes for $PM_{2.5}$ classification and information on the features used for training the classifiers are missing, so that their results are hard to compare to ours. For example, setting the number of classes to 2 and using all features of the data set (including the other pollution attributes PM_{10}, SO_2 etc.) would significantly ease the estimation task. Also, we have no information on the choice of test sets or whether cross validation was applied in the work of Hu et al. It is even possible that Hu et al. have used a slightly different data set than we have: The data set they describe includes “more than 100 000” data records from 13 meteorological stations in Beijing, while our data set contains more than 420 000 records from 12 stations.

One consistency across both their work and ours is the accuracy drop from centralized learning to FL, with 4 percentage points in [HGLM18] and 4.4 percentage points in our work.

5.3.3 Mobile radio (LTE)

MNO data is a valuable source for obtaining high-frequency and spacial insights in various fields, including population structure, mobility and the socio-economic impact of policy interventions. However, a lack of legal frameworks permitting access to data of all providers, as seen in cases like Germany, constrain the quality of analysis [SBH22]. Accessing only data of selected providers introduces biases, making FL an attractive solution to enhance the representativeness by enabling the aggregation of insights from multiple major MNOs.

Thus, our third use case is based on private MNO data owned by the company umlaut SE[12]. Different from the thirst two use cases, we had no direct access to the data, just as the aggregation party in realistic FL settings. While this allows practical insights, it also comes with economy-driven constraints regarding resources available for the simulation. Hence, the focus of this use case is more on practical engineering issues of FL and less on optimal results.

The data set in this use case contains mobile communication network coverage data, including latency and speed tests, each linked to the mobile LTE devices of individual users and a specific timestamp. The data records are also associated with GPS coordinates, such that a daily "radius of action" can be computed for each user. This radius describes how far a user has moved from their home base within one day. The user home bases have also been computed on the available data – a home base is defined as the place where most data records have been recorded. The ML task we model in this use case is to estimate the daily radius of action for a user, given different LTE metrics of one particular day (see below).

Data set The whole data set originally contains 286 329 137 data records. The following features of the data set have been aggregated for each day and user: *radius of action* in meters, *share of data records with WiFi connection* and the variance and mean values for each of the following LTE metrics: *RSRQ*, *RSRP*, *RSSNR* and *RSSI*. The date has been encoded into three numeric features (*calendar week*, *day of the week* and *month*) and the boolean feature *weekend*.

Data preprocessing We set a specific time frame of six months and a geofence around the German state of North Rhine-Westphalia, all other records are excluded – leaving 2 718 416 records in the data set. Additionally, we apply a filtering strategy to clean our data: each user in the data base needs to have data for at least 20 different days (within the time span of six months) and 10 records on each of these days. Otherwise, all records of this user are discarded. After the second filtering step, there are 1 508 102 data records in the data set, which we scale using a standard scaler implementation and then use for training, validating and testing our models.

60% of the data are used as training data, 20% are used as validation data and the remaining 20% as test data. For FL, we have divided the data set according to the mobile network operators (MNOs) of the users. Since more than 99.6% of the data records are associated with three major providers, the other 0.4% of the data records (belonging to 29 other MNOs) are eliminated from the data set.

12. umlaut website https://www.umlaut.com/, accessed on July 17, 2023

Model	R^2	Rel. loss (%)
ANN	0.130	17.7
ANN (federated)	0.114	27.8
random forest	0.158	0.0

Table 5.5: Performance in the mobile radio simulation.

Setup We use two centralized learning benchmarks: a random forest regressor and an ANN, which have both been subject to a hyperparametersearch prior to their training. The network architecture for both the centralized benchmark ANN and the FL training process is the same: The first layer consists of 28 dense-neurons and the second layer consists of 14 dense-neurons, which lead to the single-neuron output layer. All dense layers except for the output layer use the ReLu activation function. For FL, we use the SGD optimizer with a server learning rate of 3.0, a client learning rate of 0.8 and a batch size of 2.

Results The benchmarks of the centralized learning regressors are R^2 values of 0.158 (random forest), 0.13 (ANN) and 0.13 (linear regression). For the ANN trained in the FL scenario, we achieve a slightly lower R^2 value of 0.114 (see Table 5.5).

Discussion The reasons behind the weak performance of the benchmark models (R^2 of 0.158 and 0.13) are not clear. The hyperparameters might not be optimal, since we were not able to spend many resources on hyperparameter tuning due to time constraints of the data owner. Another reason might be the that the modeled task (estimating the radius of action based on LTE connection data) is inherently hard to learn. With an R^2 of 0.114, we were able to reproduce this performance in the FL setting.

Since the private data set in this use case has not left company premises, there are important lessons to be learned from a practical perspective:

1. Even if the data set is not directly available during the model engineering process, it is crucial to get basic insights on the features and statistical properties before starting the training. Crucial decisions, such as the type of model to be trained, can be made based on this.

2. Thorough HPO is crucial to obtain useful results. It might take a lot of time and computational resources to find hyperparameters which are suited for the task.

3. Technical difficulties while creating the necessary APIs and setting up the chosen ML framework at the FL clients can slow down the process even more. Without access to the data base, it might be hard to reproduce some of the errors.

While all points mentioned above were encountered in the third simulation, there was only *one* party who held all data. In real FL scenarios with multiple data holders, the process might get much more complicated.

5.4 Key Observations

Our simulations lead to the following key observations:

Models trained via FL can reach a performance very close to models trained with centralized ML approaches, as we have shown in all three use cases. While the performance gap itself is not surprising (since the FL model has been exposed to the complete data set only indirectly), we want to stress that without FL, many ML scenarios might not be possible due to privacy concerns, trade secrets, or similar reasons. This is especially true for health care data, i.e., the domain of our first simulation.

While the random forest regressor has demonstrated superior performance compared to other centralized learning benchmarks in all three simulations, exploring the potential of tree-based models within a FL context [AQKC$^+$23, YOW22, LWH20] could be a promising avenue for further investigation. The improved interpretability and explainability over many other models, e.g., ANNs, is another advantage of tree-based models.

On the other hand, random forest regressors are not suitable if tasks get more complicated. Also, their architecture, i.e., many decision trees which may be individually overfitted to parts of the training data, can facilitate the extraction of sensitive information of the training data and thus pose an additional privacy risk.

Choosing the right hyperparameters is crucial for any ML model. Since automatic HPO is still an open problem for FL algorithms, (manually) finding the right settings can be a time consuming process. Developing a suitable framework for automated HPO for FL would be important future work – although for official statistics, other issues might be more pressing at the moment (see Section 5.5).

In our third simulation (mobile radio data), we did not have access to the training and test data set, just like in a real-world scenario. This means both HPO and technical debugging needed to be performed remotely, without access to the data. Although this was already challenging, we believe that in scenarios with multiple data holders and possibly heterogeneous data sets, these tasks will be significantly harder.

All FL simulations were performed on the machine which also had access to the complete data set. In a real-world application, where each client runs on a distinct machine, other settings and other frameworks might be more practical than TensorFlow Federated.

Last but not least, we want to emphasize that FL, despite its privacy-enhancing character, may still be vulnerable to some ML privacy issues (see Section 5.2.2). Hence, analyzing these risks is a crucial step before an application is rolled out in practice. Possible mitigation strategies might be found in privacy-preserving machine learning (PPML) techniques [YZH21].

5.5 Implications for Official Statistics

In this study, we demonstrated how FL can enable NSOs to address pressing data needs in fields that are relevant to policymakers and society. Official statistics are characterised by high accuracy while underlying strict standards in confidentiality and privacy. Accuracy, explainability, reproducibility, timeliness, and cost-effectiveness are essential quality for statistical algorithms [YTB+22]. In this setting, our findings indicate that FL holds significant potential to support statistical production and improve data quality.

We showed that FL can empower NSOs to generate reliable models that accurately capture global relations. In each of our use cases, the FL-generated models exhibited nearly identical predictive performance compared to a model created by combining all available data. Each model architecture that performed well on centralized or local data could be easily adapted to a FL training process with a similar level of predictive performance only using distributed data.

If upcoming applications require to optimize an individual model for each participating party, personalized FL can be used to generate potentially improved models tailored to individual clients. This increases the interest to cooperate for each participating party, as it offers to enhance the analytic potential for each client and the server. However, it is important to note that this customization may come at the cost of global predictive performance.

FL provides the main advantage of not needing to exchange sensitive data (see Section 5.2.2). Additionally, there is no need to store or process the complete data set centralized in the NSOs.

NSOs can be empowered to appraise novel data sources sans the need for new legislation. In cases where legislative changes prove impractical, FL provides a crucial pathway to assess and prepare for regulations' modernization. By showcasing the advantages and implications of accessing new data sources before legal frameworks permit, FL not only significantly accelerates and relieves statistics production but also occasionally enables it.

To ensure successful future implementations of FL in NSOs, it is essential to focus on futher advancements. Specifically, improvements are crucial in communication frequency to enable high-speed and efficient exchanges. Our observations indicate that FL generally requires a greater number of epochs (distributed across communication rounds) compared to centralized training to achieve similar performance levels. In our use cases, even with small datasets, we found that at least 50 rounds of communication were necessary. This would result in high delay and cost in real-world applications. Therefore, the developmentof infrastructure for seamless sending and receiving ML models is necessary. Addressing this challenge, we discovered that the implementation of adaptive server optimization techniques reduced the training rounds and contributed to training stability. As a result, we recommend the use of adaptive optimizers to help minimize communication costs and enhance the efficiency of FL processes. By incorporating such adaptive optimization methods, NSOs can optimize the performance and effectiveness of FL while reducing the burden of communication overhead.

Additionally, partners need tools to update models effectively. This requires coordination of the server and expertise from all participating parties. In practice, real-world applications of FL often involve the challenge of harmonizing client data without directly accessing it. Achieving an optimized model architecture uniformly across all clients also necessitates the knowledge and collaborative efforts of the clients themselves. Providing comprehensive tools and resources to partners enables them to actively contribute to the model updating process while maintaining data privacy and security.

FL is evolving rapidly and both industry and research will continue to improve the field in the coming years. The performance and efficiency of practical FL frameworks is expected to be further optimized. Similarly, we expect the development of more usable PPML algorithms including the ones based on secure multi-party computation (SMPC) and homomorphic encryption (HE) – allowing for provably secure collaborative ML. Although such PPML methods have been proposed and frameworks exist, their performance today is often far from acceptable for many practical applications. With more standardization and simpler, respectively more efficient, applications, FL will become even more beneficial to official statistics.

In summary, FL should indeed be recognized as an important technology that can facilitate the modernization of legal frameworks for official statistics. It enables NSOs to safely use publicly relevant information that is not expected to be accessed by future legal frameworks, ultimately enhancing the quality and relevance of official statistics. However, further development is still required to fully realize the potential of FL in this context.

5.6 Conclusion

In conclusion, FL has a lot of potential for official statistics. In scenarios where external partners are unwilling to share individual-level information but still aim to analyze or disseminate global insights in their field of application, FL can help to overcome these issues. We have shown across a range of three simulated use cases that FL can reach a very similar performance to centralized learning algorithms. Hence, our results indicate that if classic (centralized) ML techniques works sufficiently, also FL can possibly produce models with similar performance.

One of the next steps to transfer FL into the practice of official statistics could be to conduct practical pilot studies. These could further showcase both the applicability and challenges of FL beyond a simulated context. Another focus of future work in this area could be the analysis of privacy risks in FL scenarios of official statistics and possible mitigation strategies. This would be an important stepping stone in ensuring the privacy protection of involved parties, even if FL itself is often already categorized as a privacy-preserving technology. Just as in countless other domains, we expect FL to become a relevant technology for official statistics in the near future.

References

[AQKC$^+$23] Mohammad Al-Quraan, Ahsan Khan, Anthony Centeno, Ahmed Zoha, Muhammad Ali Imran, and Lina Mohjazi. Fedtrees: A novel computation-communication efficient federated learning framework investigated in smart grids, 2023.

[BDF18] Martin Beck, Florian Dumpert, and Joerg Feuerhake. Machine learning in official statistics. *arXiv preprint arXiv:1812.10422*, 2018.

[DB17] Florian Dumpert and Martin Beck. Einsatz von machine-learning-verfahren in amtlichen unternehmensstatistiken. *AStA Wirtschafts-und Sozialstatistisches Archiv*, 2(11):83–106, 2017.

[HAPC17] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *ACM CCS*, pages 603–618, 2017.

[HGLM18] Binxuan Hu, Yujia Gao, Liang Liu, and Huadong Ma. Federated region-learning: An edge computing based framework for urban environment sensing. In *IEEE GLOBECOM*, pages 1–7. IEEE, 2018.

[KKP20] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

[LWH20] Qinbin Li, Zeyi Wen, and Bingsheng He. Practical federated gradient boosting decision trees. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4642–4649, 2020.

[MMR$^+$17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. arXiv, 2017.

[MSDCS19] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

[NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE SP*, pages 739–753. IEEE, 2019.

[RCZ$^+$20] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[SBH22] Younes Saidani, Sarah Bohnensteffen, and Sandra Hadam. Qualität von Mobilfunkdaten – Projekterfahrungen und Anwendungsfälle aus der amtlichen statistik. *WISTA - Wirtschaft und Statistik*, (5):55–67, 2022.

[SRS17] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, pages 587–601, 2017.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[Sto23] Joshua Stock. Federated learning simulations. https://github.com/joshua-stock/fl-official-statistics, 2023.

[YOW22] Fuki Yamamoto, Seiichi Ozawa, and Lihua Wang. efl-boost: Efficient federated learning for gradient boosting decision trees. *IEEE Access*, 10:43954–43963, 2022.

[YTB$^+$22] Wesley Yung, Siu-Ming Tam, Bart Buelens, Hugh Chipman, Florian Dumpert, Gabriele Ascari, Fabiana Rocci, Joep Burger, and InKyung Choi. A quality framework for statistical algorithms. *Statistical Journal of the IAOS*, 38(1):291–308, 2022.

[YZH21] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[ZGD$^+$17] Shuyi Zhang, Bin Guo, Anlan Dong, Jing He, Ziping Xu, and Song Xi Chen. Cautionary tales on air-quality improvement in beijing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2205):20170457, 2017.

[ZLH19] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

6 DealSecAgg: Efficient Dealer-Assisted Secure Aggregation for Federated Learning

Citation	Joshua Stock, Henry Heitmann, Janik Noel Schug, and Daniel Demmler (2024). DealSecAgg: Efficient Dealer-Assisted Secure Aggregation for Federated Learning. In *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES)*. Association for Computing Machinery, New York, NY, USA, ISBN: 979-8-4007-1718-5, Article 183, pages 1–11. DOI: 10.1145/3664476.3670873
Venue	21st International Workshop on Trust, Privacy and Security in the Digital Society (TRUSTBUS'24; co-located with ARES'24)
Ranking	**B** (CORE 2023), **B** (CORE 2021), **B** (ERA 2010)
Status	Published. This work is licensed under a Creative Commons Attribution International 4.0 License (CC BY 4.0).
Publication Type	Research Paper
Aim	The aim of this paper was to enhance the anonymity of clients in the federated learning training process. Specifically, the goal was to develop an efficient and secure protocol which creates an anonymity group around all participating clients. As a result, the aggregator should not be able to link contributions to individual clients during the model training. The protocol should not have a negative impact the utility of the trained model, it should be lightweight and require as little trust in the participating parties as possible. If clients stop responding and drop out of the training process (a common event in federated learning processes with hundreds or thousands of clients), the runtime of the protocol should not significantly increase.

Continued on next page

Methodology	The performance of the developed protocol is evaluated both theoretically and experimentally. Its runtimes are compared to related work in terms of computational complexity and through experiments. A mathematical proof for the security of the proposed protocol is also provided.
Contribution	DealSecAgg, a federated learning protocol which creates an anonymity group around all participating clients regarding their contributions, is introduced. Anonymity is achieved by masking client updates cryptographically, which preserves the resulting model performance of plain federated learning. In DealSecAgg, the masks are managed by one or more independent dealer parties, a concept borrowed from secure multi-party computation protocols. Dealers have no access to client updates or training data, thus little trust is needed. At the same time, the DealSecAgg protocol needs strictly less communication rounds than related work. This is reflected in the experimental results of this paper, where DealSecAgg takes up to 87.8% less time to run than related work. Also, clients dropping out during the training process have no negative impact on DealSecAgg's runtimes, which is a significant advantage compared to related work. The code for the experiments of this paper is available open source [Hei23].
Co-authors' contribution	Henry Heitmann wrote a master's thesis titled *Privacy-Preserving Data Aggregation for Federated Learning* (supervised by Joshua Stock) on which this paper is based. He implemented the experiments, drafted the first version of the protocol and assisted during the writing of the paper. Janik Noel Schug helped in writing the security analysis (Section 6.5) and in adjusting the quantization (Section 6.4.2). Daniel Demmler assisted in revising and refining the paper.

Abstract

Federated learning eliminates the necessity of transferring private training data and instead relies on the aggregation of model updates. Several publications on privacy attacks show how these individual model updates are vulnerable to the extraction of sensitive information. State-of-the-art secure aggregation protocols provide privacy for participating clients, yet, they are restrained by high computation and communication overhead.

We propose the efficient secure aggregation protocol *DealSecAgg*. The cryptographic scheme is based on a lightweight single-masking approach and allows the aggregation of the global model under encryption. *DealSecAgg* utilizes at least one additional dealer party to outsource the aggregation of masks and to reduce the computational complexity for mobile clients. At the same time, our protocol is scalable and resilient against client dropouts.

We provide a security proof and experimental results regarding the performance of *DealSecAgg*. The experimental evidence on the CIFAR-10 data set confirms that using our protocol, model utility remains unchanged compared to plain FL. Furthermore, the results show how our work outperforms other state-of-the-art masking strategies both in the number of communication rounds per training step and in computational costs, which grows linearly in the amount of active clients. By employing our protocol, runtimes can be reduced by up to 87.8% compared to related work.

6.1 Introduction

With federated learning (FL), machine learning models can be trained on a distributed data set in a decentralized manner [KMRR16, MMR$^+$17]. FL does not require participants to transmit their private training samples. Instead, an aggregation server distributes an initial model to the data holders, who train it locally with their private data. After each local training round, the aggregator collects the model updates and combines them, resulting in an updated global model.

While FL allows the participants to keep their data private, the exchange of model updates between aggregator and clients might still leak sensitive information (see Section 6.2.2).

Contributions We propose a simple, yet effective dealer-assisted masking strategy to mitigate the aforementioned information leakage. Our scheme *DealSecAgg* has the following properties:

- Utility: The performance of FL is not negatively impacted by our protocol; no utility loss is induced.
- Privacy: In our protocol, dealers are never exposed to actual model updates. Their domain is limited to storing, aggregating and communicating random vectors, i.e., masks.
- Security: DealSecAgg is provably secure against a semi-honest adversary unless the aggregator and all dealers are corrupted, or the aggregator and sufficiently many clients per honest dealer are corrupted.
- Efficiency: By outsourcing mask administration to the dealer party, clients and the aggregator only need to perform trivial computations in $O(1)$. The dealer's computational effort grows linearly in the number of active clients. Our protocol requires only 2 communication rounds per training epoch, which is *strictly faster than related work*.
- Minimal non-collusion requirements: We propose two flavors of the DealSecAgg protocol – a single dealer setting and a setting with multiple dealers. In the multi-dealer version, privacy guarantees are not violated as long as *one* of the dealers does not collude with the aggregator. Hence, a single trusted dealer is sufficient. Very little computational overhead is introduced in the multi-dealer scenario (since all dealers can compute in parallel) and the communication grows linearly in the number of dealers.
- Robustness: Client dropouts do not introduce any computational overhead.

Paper outline The rest of this paper is organized as follows: After a brief recap on FL and privacy attacks in Section 6.2, we provide an overview of existing secure aggregation techniques for FL in Section 6.3. In Section 6.4, we present our masking scheme DealSecAgg in the basic single dealer scenario and a multi-dealer scenario. We evaluate our scheme both theoretically in Section 6.5 and experimentally in Section 6.6, comparing its performance to three state-of-the-art masking schemes from related work. We discuss our results in Section 6.7 and draw conclusions in Section 6.8.

6.2 Background

In this section, we provide an overview of FL and relevant privacy attacks.

6.2.1 Federated Learning

In FL, a central machine learning model is trained by n data holders in a collaborative fashion. In each training round τ, the aggregation server selects a set of the data holders as clients C_τ with $|C_\tau| \leq n$ to which it sends the global model g_τ. After each client $c_i \in C_\tau$ has updated the global model per local training, they send the updated model back to the aggregator. The received updates are then combined by the aggregation server with an aggregation function to obtain the new global model for round $\tau + 1$. In this work, we use the most common aggregation function *federated averaging* (FedAvg) as suggested by McMahan et al. [MMR+17]. The global model g_τ is updated by

$$g_{\tau+1} = \sum_{c_i \in C_\tau} \frac{u_i}{|C_\tau|}, \text{ with model update } u_i \text{ of client } c_i.$$

DealSecAgg focuses on *cross-device* FL [MPP+21], which is characterized by a large number of clients (e.g., 10.000 clients in each training round [KMRR16]). In this setting, clients are often run on mobile devices, e.g., smartphones, which are powered by battery and rely on mobile networks for connectivity. Hence *dropouts*, i.e., clients failing to respond on time during an FL training process, can occur frequently and must be accounted for [MÖJC23].

6.2.2 Privacy Attacks

In the standard version of FL [MMR+17], all model updates are transferred from the clients to the aggregation server in the clear. The aggregation server and potential eavesdropper can extract sensitive information from these model updates. We briefly introduce the most important *privacy attacks* to motivate our masking strategy. Note that other threats, such as poisoning and Byzantine attacks, significantly differ in their goals and threat models and are therefore out of scope for this work [MPP+21].

Membership inference. The goal of membership inference attacks [SSSS17] is to decide whether a specific data sample was included in a model's training data. A white-box attack, where the adversary has full access to the trained weights of a model (which is the case for an FL aggregator), is more likely to reveal membership information than a black-box attack [NSH19]. The relevance of membership inference for FL is also stressed in [TLG+19], where the authors demonstrate how more variety in the training data sets among different clients yields higher attack success rates.

Property inference. Property inference attacks do not aim at specific training data samples, but at stochastic properties of the whole training data set [AMS+15, SWDF23]. A popular example are demographic properties such as the distribution of age or gender. By observing a model's weights or its behaviour, an attacker can reveal customer demographics information of a (competing) company or the distribution of host types in a computer network to refine a malware attack [GWY+18].

Data reconstruction. There are also attacks aiming at extracting concrete training samples from a trained model [BX22, RG21]. While *model inversion*, a similar attack, allows to reconstruct class representatives [FJR15], reconstruction attacks (or *data extraction*) reconstruct exact training samples. An example in the federated setting has been demonstrated in the domain of facial recognition [WSZ+19].

Most of these attacks can be performed on the trained global model after training. Attacking the clients' model updates however usually leads to more precise information, due to the direct training data exposure. Since the extracted information can be directly linked to the individual client, the promise of FL to keep the data of clients private is violated. This motivates our threat model, in which individual client updates are targeted by an attack (see Section 6.4.1).

6.3 Related Work

Before presenting our secure aggregation approach DealSecAgg, we introduce relevant techniques from related work. *Secure aggregation* is an umbrella term describing methods which create an anonymity group of all contributors during a distributed data aggregation process. Secure aggregation does not prevent the aforementioned attacks in FL alltogether, since the global model can still be attacked after training. However, it renders the linkage of extracted information to an individual client impossible and therefore protects the privacy of individual client contributions during FL training. More specifically, secure aggregation allows participants to collaboratively compute the aggregation of all updates. In order to compute this function, a cryptographic protocol is required. Various approaches have emerged that are able to provide this functionality. They can be grouped into the four categories single-masking, pairwise-masking, secure multi-party computation (SMPC) and homomorphic encryption (HE) [MÖJC23]. Compared to our proposed masking scheme, these techniques come with high computational overhead, as the following overview shows.

The focus of this work is secure aggregation in the semi-honest threat model (see Section 6.4.1). While Bonawitz et al. [BIK+17] add a consistency check to achieve security in the malicious threat model, only semi-honest versions of the protocols are mentioned throughout the rest of this paper to ensure comparability.

6.3.1 Secure Aggregation Based on Masking

In masking schemes, a pseudo-random vector of the same size as the update vector is added to the update vector — similar to a one-time pad. Therefore, it becomes infeasible to distinguish the masked vector from a random vector and no information is leaked. There are two possible procedures to unmask the aggregate: pairwise-masking and single-masking.

Pairwise-Masking First introduced by Ács and Castelluccia [ÁC11], pairwise-masking generates pairs of masks between every pair of participating clients. The idea is that every pair of clients (u, v) agrees on a shared mask m — e.g., using the Diffie-Hellman key exchange [DH76] — such that u adds and v subtracts m from their model update. Therefore, m cancels out when u and v are added. After the aggregation (summation) of all model updates, all masks cancel out. Bonawitz et al. [BIK+17] were the first to introduce secure aggregation to FL in 2017. They chose a similar approach to Ács and Castelluccia [ÁC11] using pairwise-masking.

Bell et al. [BBG+20] have enhanced this approach by limiting the amount of clients each client has to share masks with. Previously, each client had to share a mask with each other client, resulting in a complete communication graph. Bell et al. have shown that a k-regular graph is sufficiently secure and reduces the communication and computation overhead significantly. More precisely, each client shares a mask with its $O(\log n + \sigma + \eta)$ neighbors. σ and η are security parameters to ensure security with low numbers of clients and recoverability of masks. Nevertheless, with high numbers of clients, the communication graph is nearly logarithmic. Thus, less messages have to be exchanged and fewer masks must be generated.

The initial secure aggregation protocol by Bonawitz et al. [BIK+17] has also been enhanced by others. Kadhe et al. [KRKR20] introduced FastSecAgg, a protocol that uses a secret sharing scheme based on the fast Fourier transformation. While FastSecAgg lowers the computational complexity during secret sharing and mask recovery, it has lower guarantees regarding dropout resilience and colluding clients. So et al. [SGA21] proposed Turbo-Aggregate, which divides clients into groups to achieve logarithmic overhead, similar to Bell et al. [BBG+20]. Whereas in previous protocols, clients agreed on a shared seed to generate masks, this protocol requires clients to share their masked model parameters with other clients in the group. Therefore, a higher amount of data needs to be transferred and multiple rounds to share the masks between clients are needed. Similar proposals to decrease communication costs and improve dropout resilience [MWA+23, JNMALC22, JNMALC23] have been published since [SGA21]. Mansouri et al. give a good overview of the state of the art [MÖJC23].

Single-Masking As introduced by So et al. [SNY+22], single-masking allows for faster generation of masks compared to pairwise-masking. A mask-encoding that is shared between clients allows the aggregator to unmask the result in a single step. Unlike in previous pairwise-masking schemes, this approach does not require quadratic overhead when clients drop out.

In this protocol, each private mask is encoded and split into n individual masks, one of which is sent to each other client. The encoding allows to reconstruct the aggregate of all private masks of active clients, even if a fraction of clients dropped out. After the aggregator has received the masked updates, it notifies the active clients about which clients dropped out. The active clients perform an aggregation of the encoded masks they have received in the first round of the protocol, ignoring the masks of dropouts. The aggregation server can then unmask the aggregate of the models in a single step. While this protocol eliminates the overhead created by pairwise-masking, it introduces additional computational complexity in the encoding algorithm on client and server side. Additionally, just as in pairwise-masking, each client has to communicate with every other client to share its mask encoding.

6.3.2 Secure Aggregation with Secure Multi-Party Computation

Another approach to achieve secure aggregation in FL is secure multi-party computation (SMPC). One of the most celebrated SMPC proposals for FL is SAFELearn by Fereidooni et al. [FMM+21]. Their approach requires multiple aggregators to aggregate the global model using a SMPC protocol. Clients generate m secret shares of their local model update, one for each of the m aggregators. The aggregators perform a joint evaluation of the FedAvg algorithm on these shares afterwards. The aggregators do not learn anything about the model. After the aggregation, the global model is retransmitted to the clients using secret-sharing, such that clients can combine the shares to receive the plaintext global model.

The clients' overhead consists of the arithmetic secret sharing [GMW87], performed in only two rounds of communication. Yet, multiple semi-honest and non-colluding aggregators are required to achieve security guarantees. On top, SMPC protocols introduce additional overhead – even efficient protocols such as garbled circuits [Yao86] require substantial overhead to evaluate simple boolean circuits. And although FedAvg only consists of additions and one division, these need to be performed on vectors of high dimension.

Another set of SMPC-based secure aggregation protocols has been released recently by Mirval et al. with a focus on reliability and dropout resilience [MBSP23]. Furthermore, similar protocols have been improved in regards to robustness against malicious clients [KTC20], robustness against malicious aggregators [BTL+21, TLB+21] and robustness against poisoning attacks [BIMP+23]. A notable improvement on computing time has been recently proposed in [BDG+23], using GPU powered SMPC secure aggregation. However, our protocol DealSecAgg is still more efficient in comparison, due to the aforementioned overhead introduced by SMPC protocols.

6.3.3 Secure Aggregation with Homomorphic Encryption

Secure aggregation can also use homomorphic encryption (HE) to prevent the aggregator from learning anything about the private model parameters. Zhang et al. [ZLX+20] proposed BatchCrypt, an HE scheme for FL that allows the encryption of batches to reduce computation and communication overhead caused by HE. Still, the computations that are performed suffer from a bottleneck caused by runtimes. Hence, they limit their approach to the cross-silo setting, i.e., to a low number of clients with reliable connections and high computational resources. Zhang et al. train their model on nine clients. In the instance of BatchCrypt, more than 80% of the runtime is spent on encryption or decryption. Also, the amount of transferred data is 150 times greater compared to plain FedAvg.

The aforementioned protocol SAFELearn [FMM+21] is also adaptable to HE. Instead of performing an SMPC protocol between multiple aggregators, clients homomorphically encrypt their local model updates, which are evaluated by a single aggregator. They use a multi-party encryption scheme [MTPBH21] that splits the secret key between clients. Clients are able to encrypt and decrypt the model parameters, whereas the aggregator is not.

6.4 Dealer-Assisted Secure Aggregation

The overall goal of FL is to jointly compute a shared global model without the need to share private data of individuals. Naive methods to achieve this goal are not sufficiently secure due to the attacks discussed in Section 6.2.2. Cryptographic protocols have been applied to bridge this gap. However, the protocols presented in Section 6.3 create extensive computation and communication overhead. Our protocol DealSecAgg is designed to work more efficiently, while maintaining user privacy and being able to handle thousands of clients and potential dropouts.

Compared to pairwise-masking, generating masks is simpler in single-masking. However, the protocol by So et al. [SNY+22] introduces expensive computations to realize the unmasking of the aggregate. That is, clients that did not drop out need to compute the sum of all encoded masks of other surviving clients, which results in $O(n)$ operations on vectors of the same size as the model.

In this paper, another approach to single-masking is proposed. Just like in many SMPC protocols [Bea97, Bea98, DHNO19], we involve an independent dealer party. While a dealer is often used to provide randomness to parties in SMPC protocols, it is responsible for administering masks in our protocol. This drastically reduces the computations and communication performed by clients and the aggregator, without compromising the privacy of clients, as the dealer only sees random masks and never gains access to private client data.

6.4.1 Adversary Model

Our solution builds on the *semi-honest* adversary model [Gol04], meaning that an adversary follows the protocol while gathering as much information as possible. In contrast, a *malicious* adversary is allowed to deviate from the protocol, e.g., send corrupted model updates during an FL training process.

The single dealer variant of DealSecAgg in Section 6.4.3 assumes that aggregator and dealer do not collude. Since collusion is hard to rule out in practice, an extension to the protocol using multiple dealers is introduced in Section 6.4.4. This multi-dealer variant assumes that at least one dealer is not colluding with the aggregator. All other dealers may collude with the aggregator, i.e., a corrupt majority of dealers is assumed.

6.4.2 Quantization

To save bandwidth and to enable a secure masking of model updates, we propose quantization for DealSecAgg, i.e. the encoding of real numbers in integers. We use the encoding technique of [CS10], as suggested in [KS22]: A real number $x \in [-2^r; 2^r - 2^{-f}]$ is encoded as $Q^f(x) := \lfloor x \cdot 2^f \rceil \in \mathbb{Z} \cap [-2^{f+r}; 2^{f+r} - 1]$ where f is a positive integer specifying the precision and r characterized the domain of the real numbers. The decoding Q^f_{-1} works accordingly, i.e., $Q^f_{-1}(x) := x \cdot 2^{-f}$. Since this work is focused on machine learning models g_τ which contain many real numbers, we focus on vectors $x \in \mathbb{R}^{|g_\tau|}$ generalizing Q^f and Q^f_{-1}.

With the result of the encoding Q^f, we can compute on a signed ℓ-bit number, i.e., in a ring of integers modulo 2^ℓ for secure masking. Let $q_\ell : \mathbb{Z}^{|g_\tau|} \rightarrow (\mathbb{Z}_{2^\ell})^{|g_\tau|}$ denote the natural quotient map. Also, the lifting q_ℓ^{-1} works accordingly, i.e., $q_\ell^{-1}(x) \in [-2^{\ell-1}; 2^{\ell-1} - 1]$ denotes the integer y of smallest absolute value with favorism for negatives in case of equality such that $q(y) = x \in (\mathbb{Z}_{2^\ell})^{|g_\tau|}$.

Correctness of our construction requires that $\ell \geq f + r + 1 + \lceil \log_2(|C_\tau|) \rceil$ where C_τ is the set of clients the global model is sent to in round τ. Otherwise the aggregation in $(\mathbb{Z}_{2^\ell})^{|g_\tau|}$ of encoded model updates might be reduced modulo 2^ℓ. Then, its lift to $\mathbb{Z}^{|g_\tau|}$ would deviate from the aggregation of encoded model updates in $\mathbb{Z}^{|g_\tau|}$. Provided f is large enough, quantization and aggregation in $\mathbb{Z}^{|g_\tau|}$ do not degrade the model accuracy as shown in [KS22].

6.4.3 The single dealer protocol

Single-masking requires the aggregator to learn the sum of all masks to recover the aggregate of the updated model parameters (see subsection 6.3.1). In our protocol, the mask summation is calculated by a dealer, who learns nothing besides the masks. The aggregator on the other hand learns nothing but the masked model updates, as well as the global model. Therefore, none of these parties on their own are able to learn the model parameters of an individual client.

The dealer is a third party chosen by the clients. In the single dealer scenario, it is assumed that only a single dealer exists, that they are known to all clients, and that they do not collude with the aggregator. Their sole purpose is to aggregate the masks of all clients who stay active during a round of FL training. They do not have to learn the complete set of chosen clients in a training round or the tasks that the clients are performing. The dealer only learns the clients' masks and the set of clients that have stayed active throughout a training round.

The communication graph is shown in Figure 6.1. The architecture in the single dealer scenario consists of one dealer d, one aggregator a and n clients $c_i \in C$. Clients communicate with the aggregator and the dealer, but not with each other. The aggregator only learns the masked model updates of clients, but not the plaintext parameters. It communicates with the dealer to receive the aggregated masks which it needs to unmask the aggregated model.

The DealSecAgg protocol consists of two communication rounds. In the following, the first round is called the *masking phase* and the second round is called the *unmasking phase*. The aggregator holds the global model g_τ for the current training round τ. Beforehand, all parties agreed on a threshold t that defines the minimum amount of clients that must stay active throughout a training round.

To start the **masking phase**, the aggregator selects a subset of clients $C_\tau \subseteq C$ to which it sends the global model g_τ. Clients $c_i \in C_\tau$ generate model updates u_i by training g_τ with their local data. In our protocol, model updates u_i are encoded and mapped as described in Section 6.4.2. To mask the model, every client c_i selects a random key $k_i \in \{0,1\}^\kappa$. This key k_i is used as a seed for a pseudo-random generator PRG to generate a random

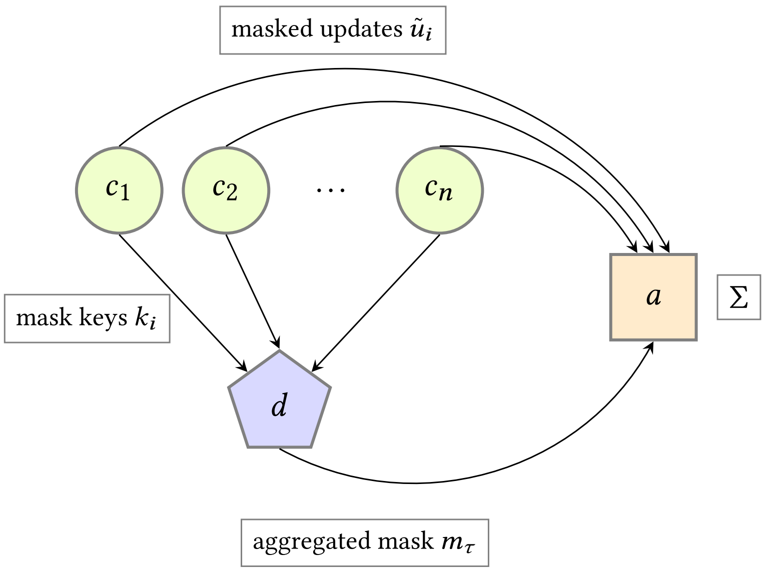


Figure 6.1: Communication between clients $c_i \in C$, dealer d and aggregator a during the aggregation and masking of one training round in the single dealer scenario.

vector m_i, the mask, which has the same size as g_τ and u_i, i.e. $m_i = \textsc{prg}(k_i)^{|g_\tau|} \in (\mathbb{Z}_{2^\ell})^{|g_\tau|}$. A masked model $\tilde{u}_i$ is computed by adding every integer-encoded value of u_i to the respective value of $m_i \in \mathbb{Z}_{2^\ell}$, i.e. $\tilde{u}_i = Q^f(q_\ell(u_i)) + m_i \in (\mathbb{Z}_{2^\ell})^{|g_\tau|}$. The key k_i is sent to the dealer and the masked model parameters $\tilde{u}_i$ are sent to the aggregator. Clients do not send their mask m_i but their key k_i to the dealer to save bandwidth. Therefore, the size κ of the key space $\{0,1\}^\kappa$ should be chosen such that it is large enough to fulfill security requirements but significantly smaller than the size of the model. A dealer can compute the mask from a client key k_i by instantiating the PRG with k_i. The aggregator waits until all clients $c_i \in C_\tau$ have sent their $\tilde{u}_i$, or, if clients dropped out, until a predefined timeout interval is reached. All clients that respond with $\tilde{u}_i$ are considered active clients $C_\alpha \subseteq C_\tau$.

In the **unmasking phase**, the aggregator sends a list of active clients C_α, as well as the size of the global model $|g_\tau|$ to the dealer. The value of $|g_\tau|$ is sent to the dealer, such that the size of generated masks is correct. The dealer then checks whether the amount of active clients is sufficient, i.e., $|C_\alpha| \geq t$, and aborts otherwise. This prevents the aggregator from requesting the mask of a single client (see Section 6.4.3). Subsequently, the dealer computes the aggregated mask m_τ for the current training round τ. For every $c_i \in C_\alpha$, it reconstructs the private mask m_i. As the dealer received the keys of all active clients, it is able to compute $m_i = \textsc{prg}(k_i)^{|g_\tau|} \in (\mathbb{Z}_{2^\ell})^{|g_\tau|}$ for each client. All of these masks are summed up to compute the aggregated mask of active clients m_τ, such that $m_\tau = \sum_{c_i \in C_\alpha}(m_i) \in (\mathbb{Z}_{2^\ell})^{|g_\tau|}$. The aggregated mask m_τ is sent back to the aggregator.

The aggregator is now able to compute the global model $g_{\tau+1}$ for the next training round, according to FedAvg. Since $\tilde{u}$ contains exactly the same masks as m_τ, the subtraction of m_τ uncovers the sum of all model parameters u_{sum}, which can be decoded back to the domain of real numbers (see functions q_ℓ^{-1} and Q_{-1}^f in Section 6.4.2). To average the parameters and receive the global model, $Q_{-1}^f(q_\ell^{-1}(u_{sum})) \in \mathbb{R}^{|g_\tau|}$ is divided by the amount of models that were summed. Hence, the aggregator needs to perform the following at the end of each round:

$$u_{sum} = \sum_{c_i \in C_\alpha} \tilde{u}_i - m_\tau \in (\mathbb{Z}_{2^\ell})^{|g_\tau|} \tag{6.1}$$

$$g_{\tau+1} = \frac{Q_{-1}^f(q_\ell^{-1}(u_{sum}))}{|C_\alpha|} \in \mathbb{R}^{|g_\tau|} \tag{6.2}$$

To initiate the next round $\tau+1$, the aggregator broadcasts the new global model $g_{\tau+1}$ to all clients in $C_{\tau+1}$.

A note on parameter t

It is important to state that $|C_\alpha| \geq t$. Otherwise, the aggregator would be able to select $|C_\alpha| = 1$, which would leak the unmasked model parameters of a single client after the unmasking phase. If $t = 2$, at least two model updates are aggregated into the global model. In practice, t should be selected high enough to prevent information leakage over multiple rounds. With only a few active clients in each round, the aggregator might be able to run a variant of a set intersection attack on the model parameters to learn which parameters were trained by which client. Bonawitz et al. [BIK+17] observed an average dropout-rate of around 10% in a mobile setting. Therefore, if t was selected to allow 15% or 20% of dropouts, high security would be achieved without risking the need to abort due to a lack of active clients.

6.4.4 The multi-dealer protocol

The main pitfall of the single dealer design is the assumption that the dealer does not collude with the aggregator, since collusion between these two parties would uncover all private model parameters. However, there is a way to allow collusion without sacrificing security. This requires multiple dealers.

DealSecAgg protocol variant	Client c_i	Aggregator	Dealer(s)
single dealer	choose key k_i, mask update u_i with $\text{PRG}(k_i)$, send k_i to dealer	request aggregated masks m_τ from dealer, compute $g_{\tau+1}$ according to Equations 6.1 and 6.2	generate m_τ and send to aggregator
multi-dealer	choose multiple keys K_i such that $\lvert K_i\rvert = \lvert D^i\rvert \leq p$, mask update u_i with $\sum_{k_i \in K_i} \text{PRG}(k_i)^{\lvert g_\tau\rvert}$, send each $k_i \in K_i$ to different dealer in D^i	request aggregated mask m_τ from all active dealers in D_α, compute $g_{\tau+1}$ according to Equations 6.3 and 6.2	generate m_τ and send to aggregator

Table 6.2: A summary of the two protocol variants single dealer (see Section 6.4.3) and multi-dealer (see Section 6.4.4).

In the multi-dealer setting of DealSecAgg, each client c_i selects a set of dealers $D^i \subseteq D$. Instead of masking the model parameters once, clients mask the parameters up to p times in total, once for each of the dealers in D^i, using a different key k_i for each mask. The dealers D^i are carefully selected by the clients from a pool of trusted dealers D with $|D| = p$. Dealers could be provided by independent organizations in different locations to minimize the risk of collusion. The communication graph is shown in Figure 6.2. Clients are free in the choice of their dealer(s), the number of chosen dealers can also vary between clients. However, to be able to aggregate a sufficient amount of masks, dealers must be chosen such that each dealer receives enough keys k_i. If K_j are the keys that a dealer $d_j \in D$ received, then $|K_j| \geq t$ must be satisfied.

Each client $c_i \in C_\alpha$ notifies the aggregator about the dealers $D^i \subseteq D$ it used. $D_\alpha = \bigcup_{c_i \in C_\alpha} D^i$ therefore is the set of all dealers that are active in the current training round. The aggregator is able to request the aggregated masks from each of these dealers. After receiving all aggregated masks m^j_τ from the active dealers $d_j \in D_\alpha$, they are subtracted from the aggregated masked models $\tilde{u}_i$ to receive u_{sum}:

$$u_{sum} = \sum_{c_i \in C_\alpha} \tilde{u}_i - \sum_{d_j \in D_\alpha} m^j_\tau \in (\mathbb{Z}_{2^\ell})^{|g_\tau|} \tag{6.3}$$

After lifting and decoding u_{sum} to a vector in $\mathbb{R}^{|g_\tau|}$, the average over the number of active clients $|C_\alpha|$ yields the new global model $g_{\tau+1}$. This calculation is identical to the single dealer protocol, see Equation 6.2. Note that the only difference between Equation 6.3

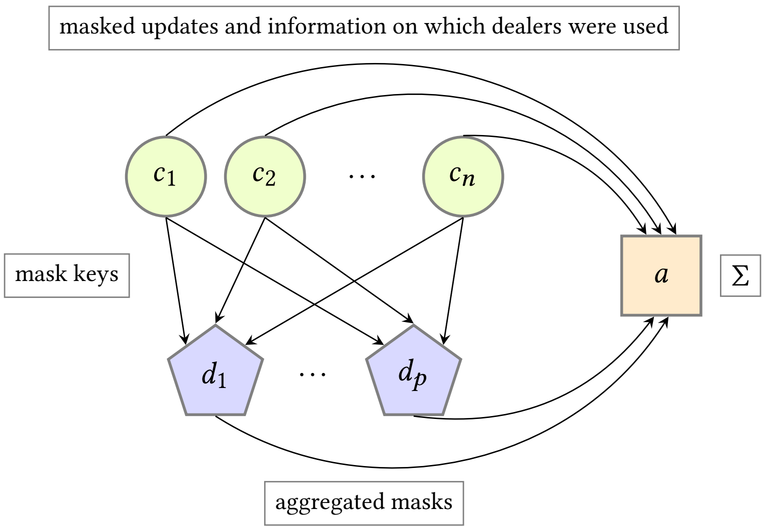


Figure 6.2: The communication between clients $c_i \in C$, multiple dealers $d_i \in D$ and aggregator a during the aggregation and masking of one training round.

and Equation 6.1 (the single dealer setting) is the aggregation of the received masks m_τ^j. An overview of the differences of our two protocol variants is provided in Table 6.2.

In the multi-dealer variant of the protocol, $p-1$ dealers are allowed to collude with the aggregator. With the aggregated model parameters being hidden by p masks, the removal of $p-1$ masks would reveal nothing about the parameters. Therefore, choosing multiple dealers can drastically reduce the risk of private data being leaked, since only one of the dealers has to be non-colluding with the aggregator. Also, the total runtime only increases slightly, as all dealers can compute in parallel and additional operations on the clients and the aggregator are negligible.

6.5 Theoretical Analysis

This section evaluates the security and efficiency of the DealSecAgg protocol.

6.5.1 Security Analysis

In our analysis, we consider the security of the multi-dealer protocol, with the single-dealer protocol being a special case. The secure aggregation protocol should provide privacy for clients in regard to their private model parameters. The model parameters that the aggregator receives are masked using the additive masking scheme defined in Section 6.4.4. The masking scheme is a generalization of the one-time pad (OTP) encryption with pseudorandomness from $(\mathbb{Z}_2)^{|g_\tau|}$ to $(\mathbb{Z}_{2^\ell})^{|g_\tau|}$, i.e., a mask of the same size as the plaintext, which is chosen pseudorandomly, is added. A security proof of the OTP with pseudorandomness was given by Katz and Lindell [KL07, Theorem 3.18], building upon [KL07, Theorem 2.9] . This proof can be adapted to the additive masking scheme and results in the following theorem:

Theorem 1. *Additive masking has computationally indistinguishable encryptions in the presence of an eavesdropper.*

Intuitively, encrypting a model with a mask of the same size, which is chosen pseudorandomly, results in a masked model that is computationally indistinguishable from a different masked model. This idea is used to show that executing the DealSecAgg protocol is secure against a semi-honest adversary unless the aggregator and all dealers are corrupted or the aggregator and sufficiently many clients per honest dealer are corrupted. Executing a protocol, the view of a party (client, dealer, aggregator) consists of its input, its random tape, its received messages and all computation results by this party. Because the results can be computed using the inputs, the view of a party does not include results.

Theorem 2. *Let V' be the collection of...*

- *...the old global model g_τ,*
- *...the local updates u_i of corrupted clients $c_i \in C_\tau$,*
- *...information on which dealers are chosen, which clients are active and which parties are corrupted,*
- *...the random tape of corrupted parties.*

If the aggregator is honest let $V'_P = V'$. If the aggregator is corrupted let $t_{honest} \in \mathbb{N}$ and let $C_{>0}$ denote the set of honest clients c_i for which there is an honest dealer $d \in D^i$ with $|\{c_i \in C \mid c_i \text{ is honest}, d \in D^i\}| \geq t_{honest}$. In the case of a corrupted aggregator, for a partition $\{P_w\}_{w \in W}$ of $C_{>0}$ with index set W let V'_P be the collection of ...

- *...the collection V' above,*
- *...the new global model $g_{\tau+1}$,*
- *...the local updates u_i of all honest clients $c_i \in C_\tau \setminus \{C_{>0}\}$,*
- *...the sums $\{\sum_{c_i \in P_w} q_\ell(Q^f(u_i))\}_{w \in W}$.*

Let F be a multivariate function. Then, for each semi-honest adversary $\mathscr{A}$ receiving the collection V of views of corrupted parties in the DealSecAgg protocol, there is a partition $\{P_w\}_{w\in W}$ of $C_{>0}$ with $|P_w| \geq t_{honest}$ for all $w \in W$ and an adversary $\mathscr{A}'$ receiving V_P' such that there is negligibility in κ of

$$\left|\Pr[\mathscr{A}(1^\kappa, V) = F((u_i)_{c_i\in C})] - \Pr[\mathscr{A}'(1^\kappa, V_P') = F((u_i)_{c_i\in C})]\right|.$$

Proof. First, assume an honest aggregator. Then V consists of V_P', all keys k_i^j for a corrupted dealer d_j, all masks m_i^j for a corrupted dealer d_j, and all sums of masks m_τ^j for a corrupted dealer d_j.

Construct $\mathscr{A}'$ as follows: $\mathscr{A}'$ samples the keys k_i^j of corrupted clients according to their random tape and the keys of honest clients uniformly at random. Then, $\mathscr{A}'$ computes $m_i^j = \mathrm{PRG}(k_i^j)$ and $m_\tau^j = \sum_{d_j \in D^i} m_i^j$. Let V_{art} denote the resulting artifical view of corrupted parties. Finally, $\mathscr{A}'$ calls $\mathscr{A}$ as a subroutine on input 1^κ and V_{art}. So,

$$\mathscr{A}(1^\kappa, V_{\mathrm{art}}) = \mathscr{A}'(1^\kappa, V_P'). \tag{6.4}$$

The only difference between V and V_{art} is who chooses k_i^j uniformly at random and hence

$$\Pr[\mathscr{A}(1^\kappa, V) = F((u_i)_{c_i\in C})] = \Pr[\mathscr{A}(1^\kappa, V_{\mathrm{art}}) = F((u_i)_{c_i\in C})]. \tag{6.5}$$

Equation 6.4 and 6.5 show the statement for an honest aggregator.

Now, assume that the aggregator is corrupted. Then V consists of V_P' except for the sums $\{\sum_{c_i\in P_w} q_\ell(Q^f(u_i))\}_{w\in W}$ together with all keys k_i^j sent to corrupted dealers d_j, the respective masks m_i^j, all sums of masks m_τ^j, and all masked updates $\tilde{u}_i$. We complete the proof in three steps. First, we construct a partition of $C_{>0}$. Then, we reduce the problem s.t. we can use Theorem 1 in the last step.

We construct the partition of $C_{>0}$ inductively. For each $n \in \mathbb{N}$ with $n \leq p$, let C_n denote the set of honest clients c_i for which there are exactly n honest dealers $d \in D^i$ with $|\{c_{i'} \in C \mid d \in D^{i'}\}| \geq t_{\mathrm{honest}}$. Start with $n = 1$ and $W_0 = \emptyset$. For $n > 0$, set $W_n := W_{n-1}$. Pick $c_i \in C_n \setminus (\bigcup_{w\in W_n} P_w)$ and choose a dealer $d \in \{d \in D \mid d \in D^i\}$. Let $W_n := W_n \cup \{d\}$ and $P_d := \{c_i \in C_{>0} : d \in D^i\}$. Repeat this process picking c_i if possible, else increment n and repeat this process. Stop if $n = p+1$.

Now, let $V_P'' = V_P$ except for m_τ^j with $d_j \in W$ but including $\{\sum_{c_i\in P_w} q_\ell(Q^f(u_i))\}_{w\in W}$. Construct $\mathscr{A}''$ like this: On input V_P'', $\mathscr{A}''$ computes $m_\tau^j = \sum_{c_i\in P_{d_j}} \tilde{u}_i - \sum_{c_i\in P_w} q_\ell(Q^f(u_i)) -$

$\sum_{d_j \in D_\alpha \setminus W} m_\tau^j$. Let V_{art} be the resulting artificial view of corrupted parties. Finally, $\mathscr{A}''$ calls $\mathscr{A}$ as a subroutine on input 1^κ and V_{art}. So,

$$\mathscr{A}(1^\kappa, V_{\text{art}}) = \mathscr{A}''(1^\kappa, V_P'') \tag{6.6}$$

and

$$\Pr[\mathscr{A}(1^\kappa, V) = F((u_i)_{c_i \in C})] = \Pr[\mathscr{A}(1^\kappa, V_{\text{art}}) = F((u_i)_{c_i \in C})]. \tag{6.7}$$

Now, construct $\mathscr{A}'$ as follows: $\mathscr{A}'$ samples k_i^j and computes m_i^j as in the case of the aggregator being honest. For corrupted clients and honest clients $c_i \in C_\tau \setminus C_{>0}$ set $\tilde{u}_i := q_\ell(Q^f(u_i)) + m_i$. It remains to construct $\tilde{u}_i$ for all $c_i \in C_{>0}$. For each $w \in W$ choose $c_{i_w} \in P_w$. For each $c_i \in P_w \setminus \{c_{i_w}\}$ set $u_i^{\text{art}} = 0 \in (\mathbb{Z}_{2^\ell})^{|g_\tau|}$ and set $u_{i_w}^{\text{art}} := \sum_{c_i \in P_w} q_\ell(Q^f(u_i))$. For each $c_i \in C_{>0}$, set $\tilde{u}_i := u_i^{\text{art}} + m_i$. Let V_{art}'' denote the resulting artifical view of corrupted parties. Finally, $\mathscr{A}'$ calls $\mathscr{A}''$ as a subroutine on input 1^κ and V_{art}''. So,

$$\mathscr{A}''(1^\kappa, V_{\text{art}}'') = \mathscr{A}'(1^\kappa, V_P'). \tag{6.8}$$

Note that in V_{art}'', the masks m_i^j of clients $c_i \in C_{>0}$ to dealers $d_j \in W$ appear exactly once, namely in masking $\tilde{u}_i$. This was the reason for the construction of $\mathscr{A}''$. Then, using the indistinguishability hop lemma [Sho04] we can apply Theorem 1 yielding that

$$\left|\Pr[\mathscr{A}''(1^\kappa, V_P'') = F((u_i)_{c_i \in C})] - \Pr[\mathscr{A}''(1^\kappa, V_{\text{art}}'') = F((u_i)_{c_i \in C})]\right| \tag{6.9}$$

is negligible. Equation 6.6, 6.7, 6.8 and 6.9 show the statement for a corrupted aggregator. □

This shows that DealSecAgg is as secure as handing out only V_P' to an adversary for a partition $\{P_w\}_{w \in W}$ of $C_{>0}$. If t_{honest} is large enough and $C_{>0}$ contains all honest clients in C_τ, this information is no more helpful to the adversary than V' together with $g_{\tau+1}$. So in this case, DealSecAgg is secure even if the aggregator and multiple dealers are corrupt.

6.5.2 Performance Analysis

In this section, the DealSecAgg protocol is analyzed in terms of its runtime and communication complexity. A summary is depicted in Table 6.3. Computational complexity is reduced to the number of vectors that need to be expanded from a seed, the most expensive computation in a masking scheme.

	Client	Dealer	Aggregator
Masking phase (round 1)	$O(1)$	—	$O(1)$
Unmasking phase (round 2)	—	$O(n-d)$	$O(1)$

Table 6.3: Comparison of computation costs of the DealSecAgg protocol in the two communication rounds. n is the number of clients participating in a training round and d is the number of dropouts.

Protocol	Rounds	Client	Aggregator	Dealer
Bonawitz et al. [BIK+17]	4	$O(n)$	$O((1-d)\,n+d\,n^2)$	–
Bell et al. [BBG+20]	4	$O(\log n)$	$O((1-d)\,n+d\,n\,\log n)$	–
Kadhe et al. [KRKR20]	3	$O(\log n)$	$O(\log n)$	–
So et al. [SGA21]	$n/\log n$	$O(\log n)$	$O(n)$	–
So et al. [SNY+22]	3	$O(1)$	$O(1)$	–
DealSecAgg (ours)	2	$O(1)$	$O(1)$	$O(n-d)$

Table 6.4: Comparison of communication and computation costs of various masking protocols with number of participating clients n and number of dropouts d.

The main computational complexity of the protocol is the generation of masks using a PRG. Yet, in the DealSecAgg single dealer protocol, only one mask has to be generated by each client. As apparent in Table 6.4, this is a major benefit compared to related work. Additionally, masks have to be generated by the dealer(s). More precisely, a dealer has to generate one mask for every client that did not drop out. For the number of dropouts d and the number of total clients n, up to $n-d$ masks are generated per dealer. The aggregator does not generate masks, it has to perform m subtractions on the masks (one for each of the m dealers). However, this computation is cheap in comparison to the mask generation.

Communication consists of two rounds: training and masking (1), and unmasking (2). Most of the message size in the first round consists of the global model that is sent to the clients and the masked model sent back to the aggregator. Masking the model does not add any bits to the model size. Also, clients send their key(s) to the dealer(s). The key has a variable size, however, a 128 bit key was used in the simulation. This is the only communication overhead compared to plain FedAvg. In the unmasking phase, the aggregator sends the model size (16 bits) and a list of active clients (16n bits) to the dealer(s). Subsequently, the dealer(s) send(s) a single mask to the aggregator. If the mask has 100,000 parameters of `int32`, then the message would have the size of 3,200,000 bits. In comparison, the aggregator has to send a model to every client. Therefore, sending a single mask per dealer does not add significant overhead.

6.6 Experiments

Simulations were performed to further evaluate the efficiency of DealSecAgg. The source code of the experiments is available on Github [Hei23].

6.6.1 Simulation Setup

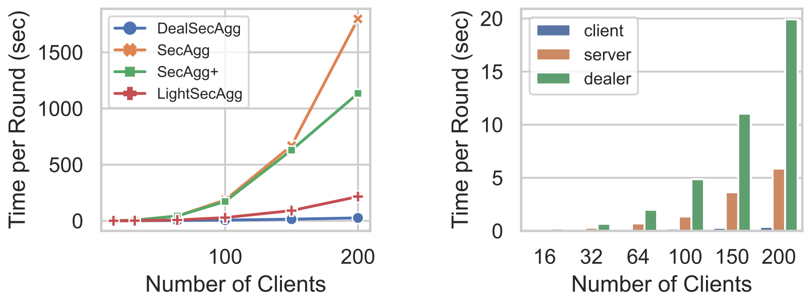
The *DealSecAgg* protocol is compared to the protocols *SecAgg* by Bonawitz et al. [BIK+17], *SecAgg+* by Bell et al. [BBG+20] and *LightSecAgg* by So et al. [SNY+22]. All protocols were initialized with a minimum number of active clients $t = \lceil n/2 \rceil$. For SecAgg+, the recommended parameters were used: $\sigma = 40$, $\gamma = 0.3$, $\delta = 0.2$ and $\eta = 30$ [BBG+20].

In our experiments, we have not implemented the encoding and decoding of real values as described in Section 6.4.2. Instead, we have directly operated on floating-point values, both for model parameters and masks, similar to the authors of *SecAgg* [BIK+17]. We expect the impact of utilizing quantization on runtimes to be negligible and focus our experiments on the computational impact of the cryptographic functions introduced by DealSecAgg.

The protocols were implemented in Python using the FedML framework [HLS+20]. It uses the *Message Passing Interface* to enable clients to communicate on multiple threads, and *PyTorch* for training the model parameters. Simulations were carried out by two Intel Xeon E5-2690 processors with a total of 32 threads. The Debian Linux 9.5 system had a total of 384 GiB of DDR3 memory. Also the training of the parameters was performed by the CPU.

6.6.2 Results

This section analyzes the runtimes, the bits that were transmitted between parties and the performance of the learned model. The main aspect that has been investigated is the scalability of the protocol, that is, how the number of clients, dropouts, and dealers affect the runtime. Furthermore, the runtime of each device is measured in order to better understand when and where expensive computations are performed. A special emphasis is put on client devices which are often mobile and therefore resource constrained. The displayed runtimes do not include the time that was used to train the parameters. Only



(a) Varying number of clients without dropouts (b) Runtime of each device for DealSecAgg

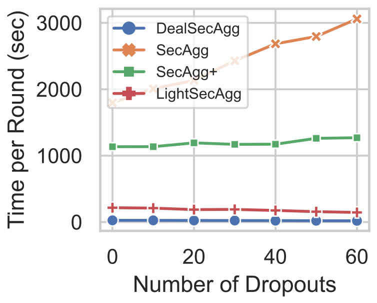
Figure 6.3: Comparison of runtimes for a varying number of clients.

the runtimes for operations that are introduced by the secure aggregation protocols are considered.

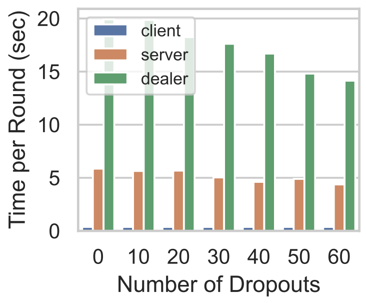
To determine the load on the network, we have tracked the amount of bits that are transmitted among FL parties. This is the basis of an evaluation in different bandwidth and latency settings.

Number of Clients The first set of analyses examines the impact of the number of clients on the total runtime of one training round. Figure 6.3 depicts a comparison of DealSecAgg to related work in (a) and the runtimes per party in (b). For this experiment, the DealSecAgg protocol operated in the single dealer setting. The former figure provides an overview of the total runtime of one training round for 8 to 200 clients. It validates the runtime complexities that have been presented in Table 6.3 and Table 6.4. The DealSecAgg protocol significantly reduces the runtime by 97.6% in comparison to SecAgg+ and 87.8% in comparison to LightSecAgg for 200 clients. With 200 clients, one training round took 27 seconds in total for DealSecAgg. As shown in Figure 6.3b, the majority of the time is spent computing the aggregated mask at the dealer, followed by the aggregation of the model parameters at the aggregator. Due to the reduced complexity of computations on the clients, they only need 0.3 seconds for computing the mask and masking the parameters.

Number of Dropouts Further analysis shows that the runtime for the DealSecAgg protocol decreases with an increase in dropouts. Figure 6.4a compares the runtime of the protocols in settings with different numbers of dropouts. In the simulation, clients drop out when they are expected to send their mask to the dealer and their masked model to the aggregator. As expected, SecAgg and SecAgg+ perform worse with an increase in dropouts due to an inefficient mask recovery process. LightSecAgg shows a slight

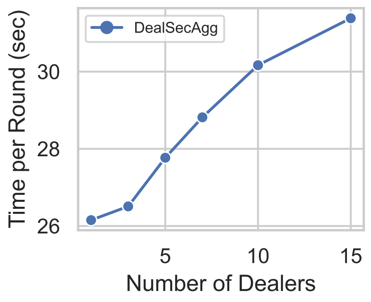


(a) Varying number of dropouts

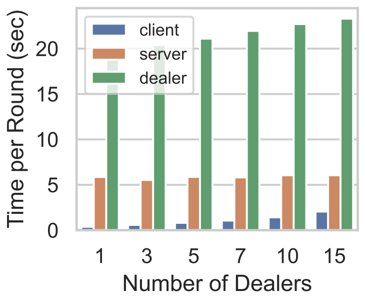


(b) Runtime of each device for DealSecAgg

Figure 6.4: Comparison of runtimes for a varying number of dropouts. 200 clients participated in total.



(a) Varying number of dealers



(b) Runtime of each device

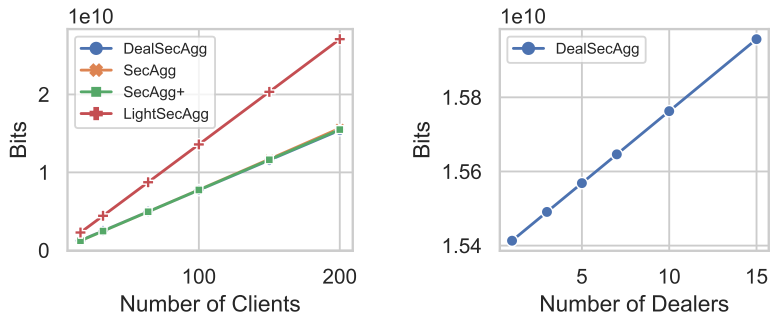
Figure 6.5: Comparison of runtimes for a varying number of dealers in the multi-dealer variant of DealSecAgg. 200 clients participated in total.

decrease in runtime due to their use of the mask-encoding. However, DealSecAgg still has a 87.3% shorter runtime compared to LightSecAgg with 60 dropouts (30% of all clients). In Figure 6.4b, the runtime of each device is shown. From this figure, it can be seen that the runtime of the dealer decreases the most. Since it has to generate fewer masks with fewer active clients, its runtime decreases. Additionally, the aggregator has to aggregate fewer model parameters.

Number of dealers An inspection of the experimental evidence on efficiency in the multi-dealer setting of DealSecAgg is depicted in Figure 6.5, illustrating the total runtimes per round with 1 to 15 dealers. Figure 6.5a shows a slight increase in runtime when the number of dealers is increased. To be precise, the runtime increases from 27 seconds with one dealer to 31.3 seconds with 15 dealers. This is still a decrease in runtime of 85.8% in comparison to the LightSecAgg protocol. In this instance, the runtime mainly increases

		DealSec-Agg (ours)	SecAgg	Sec-Agg+	LightSec-Agg
Round 1	Aggr.	916.386	944.045	927.120	1841.151
	Client	4.582	4.605	4.591	9.206
	Dealer	0	—	—	—
Round 2	Aggr.	0.017	1.083	0.432	1.083
	Client	0	0.008	0.003	0.023
	Dealer	4.582	—	—	—

Table 6.5: Comparison of communication costs in MiB sent by each party in the single-dealer setting with 200 clients and no dropouts.

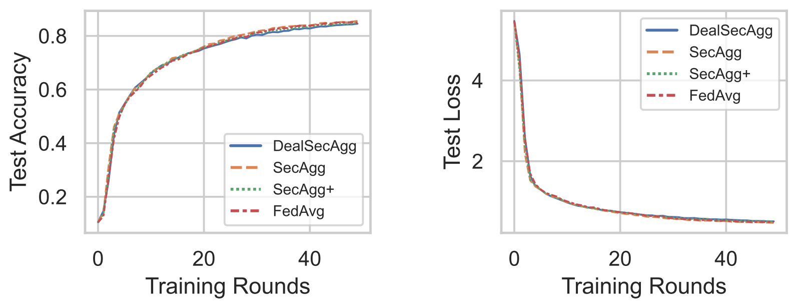


(a) Varying number of clients (b) Varying number of dealers and 200 clients

Figure 6.6: Comparison of communication in total bits sent in each training round.

at the clients and the dealer as shown in Figure 6.5b. This is the result of multiple masks being generated and added to the model parameters. However, the runtime of the aggregation of the global model is not affected by an increase of dealers. Overall, it is apparent that an increase in the number of dealers and therefore in security has comparatively little effect on the total runtime.

Communication The next section of the evaluation is concerned with the communication overhead that is induced by DealSecAgg. The main observation is that most of the transmitted data is made up of the model itself. The total amount of bits that are sent is shown in Figure 6.6a. It shows that the protocols DealSecAgg, SecAgg and SecAgg+ almost send the identical amount of bits. This can be explained by the relatively small induced overhead compared to the actual model size. Table 6.5 shows the amount of transferred bits by each device in the masking and the unmasking phase. With a model



(a) Test accuracy per training round (b) Test loss per training round

Figure 6.7: Test accuracy and loss on the CIFAR-10 dataset for each training round of various secure aggregation protocols in comparison to plain FedAvg.

size of 4.582 MiB, the aggregator sends 200× this amount in the masking phase, one to each client. Subsequently, the clients send the trained model back. Little overhead is generated in SecAgg and SecAgg+ for the secret shares, and in DealSecAgg for the key that is sent to the dealer. In the DealSecAgg protocol, the aggregated mask in the size of the model is sent to the aggregator. Compared to the total amount of transferred bits, this is rather insignificant. Significantly more data is transferred in the LightSecAgg protocol, because each client sends the encoded mask besides the masked model, amounting to twice the size of the actual model. Figure 6.6b shows the effect of multiple dealers: Clients have to send a key to each of the dealers and those have to send their aggregated mask to the aggregator. This results in a slight increase in transmitted bits with an increasing amount of dealers.

Model Performance In the final part of the evaluation, a comparison of the model performance is presented. A *resnet56* was trained on the CIFAR-10 dataset over 50 rounds. Clients trained 10 epochs locally with SGD and a learning rate of 0.02. Each client received a non-overlapping partition of the training set. Figure 6.7 shows the test accuracy: Overall, all protocols performed similarly, as expected. Slight deviations in the graph are due to the different training sets of clients. As the masking does not change any model parameters, the model performance is not affected. Both plain FedAvg and DealSecAgg, which also uses the FedAvg algorithm, achieve a test accuracy of around 85% after 50 training rounds.

6.7 Discussion

Secure aggregation was introduced to protect FL against various attacks on individual model parameters. Unlike other approaches like differential privacy [Dwo08], masking does not have a negative influence on the training performance. Hence, the resulting model is just as accurate as a model that is trained without secure aggregation. With DealSecAgg, a protocol is proposed that aims to improve efficiency while guaranteeing a similar level of security as previous secure aggregation protocols, i.e., SecAgg [BIK+17], SecAgg+ [BBG+20] and LightSecAgg [SNY+22].

Regarding computational overhead, the amount of computations has been reduced to the generation and addition of a single mask on the client side – the least amount of operations necessary to perform a masking scheme. In contrast, SecAgg requires n and SecAgg+ requires $\log n$ generations of masks for each client. The aggregator needs one subtraction to unmask the global model – also the least amount of operations that need to be performed in single-masking. All these operations are performed in constant time. In contrast on the dealer side, computing the aggregated mask requires n generations and additions of masks for each of the n clients. Therefore, the computational overhead grows linearly in the number of active clients. In comparison to SecAgg, which grows quadratically, and SecAgg+, which grows logarithmically, this is a great improvement.

When it comes to communication, the amount of bits sent between parties is similar to SecAgg and SecAgg+. Yet, while those protocols require four communication rounds, DealSecAgg requires only two rounds. This gives our protocol an advantage in networks with a high latency. Although LightSecAgg performs only slightly worse than DealSecAgg, it also introduces a high communication overhead. In practice, with limiting network bandwidth and latency, this overhead would be much more significant.

Unlike other masking-based protocols, DealSecAgg requires a dealer party. Although trusted third parties are common in SMPC protocols, e.g., to outsource operations, it might be difficult to select a trustworthy party. DealSecAgg minimizes the dealer's input, hence limiting the possibilities of a malicious dealer. Although a dealer is not able to obtain any sensitive information, they can still obfuscate the global model by sending a tampered aggregated mask.

As in most cryptographic protocols, colluding parties are a risk in DealSecAgg, as they could compromise individual model updates. By choosing the dealer carefully, the probability of a collusion can be reduced. Furthermore, incorporating multiple dealers increases the trust in the system, as it reduces the trust required for individual dealers: It suffices to have *one* dealer who does not collude with the aggregator to preserve privacy. The results of the simulation show that an increase in dealers only slightly increases the runtime of the protocol.

6.7.1 Future Work

Our main assumption is that all parties are semi-honest, i.e., that all parties follow the protocol as defined. However, an adaption of DealSecAgg to the malicious adversary model could be possible. For example, Bonawitz et al. [BIK$^+$17] propose an extension to SecAgg that protects against malicious aggregators. They add an additional communication round with a consistency check to ensure that the aggregator follows the protocol. Applying this approach to DealSecAgg would be an interesting direction for future work.

Another direction could be to assess whether DealSecAgg can be extended to protect against malicious *clients*, who could cause privacy and security risks or affect the global model through model poisoning. Approaches making FL robust against corrupted client updates could also be applicable to our protocol [LCW$^+$20, PKH22].

Extending our protocol to other aggregation functions than FedAvg is left to future work as well. Weighted averaging can be implemented as long as the aggregator informs the clients about their individual weight at the beginning of each training round. Then, clients can scale their update u_i with their assigned weight *before* encoding and masking it. At the end of each round, the aggregator has to account for clients who have dropped out by scaling the unmasked updates accordingly.

Last but not least, it could be valuable to extend our experiments to different settings for encoding model updates, as described in Section 6.4.2. Comparing model performance and communication costs for different values ℓ could provide more insights into the performance-utility tradeoff.

6.8 Conclusion

In this work, we have presented DealSecAgg, a lightweight single-masking protocol for secure aggregation in FL. We have improved on previous secure aggregation protocols [BIK$^+$17, BBG$^+$20, SNY$^+$22], which either introduce high computation and communication overhead or are not able to efficiently handle dropouts. In networks with thousands of clients, these protocols are heavily limited by their runtime. In contrast, our findings clearly indicate that in large scale cross-device scenarios, DealSecAgg drastically improves the runtime. By outsourcing the unmasking process to one or multiple dealers, the computation at the clients and the communication overhead are reduced to a minimum. While the single dealer setting requires a non-colluding third party, we provide an extension for multiple dealers in which all but one dealers may be untrusted. We have proven the security of both scenarios and have shown how the utility of the resulting FL model remains unaffected.

References

[ÁC11] Gergely Ács and Claude Castelluccia. I have a DREAM! (DiffeRentially privatE smArt metering). In *Information Hiding*, pages 118–132. Springer, 2011.

[AMS+15] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[BBG+20] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

[BDG+23] Mariya Georgieva Belorgey, Sofia Dandjee, Nicolas Gama, Dimitar Jetchev, and Dmitry Mikushin. Falkor: Federated learning secure aggregation powered by aes-ctr gpu implementation. *Cryptology ePrint Archive*, 2023.

[Bea97] Donald Beaver. Commodity-based cryptography (extended abstract). In *Symposium on Theory of computing*, pages 446–455. ACM, 1997.

[Bea98] Donald Beaver. Server-assisted cryptography. In *Workshop on New security paradigms*, pages 92–106. ACM, 1998.

[BIK+17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, pages 1175–1191. ACM, 2017.

[BIMP+23] Yaniv Ben-Itzhak, Helen Möllering, Benny Pinkas, Thomas Schneider, Ajith Suresh, Oleksandr Tkachenko, Shay Vargaftik, Christian Weinert, Hossein Yalame, and Avishay Yanai. Scionfl: Efficient and robust secure quantized aggregation. *Cryptology ePrint Archive*, 2023.

[BTL+21] Carlo Brunetta, Georgia Tsaloli, Bei Liang, Gustavo Banegas, and Aikaterini Mitrokotsa. Non-interactive, secure verifiable aggregation for decentralized, privacy-preserving learning. In *Information Security and Privacy*, pages 510–528. Springer, 2021.

[BX22] Nathalie Baracaldo and Runhua Xu. Protecting against data leakage in federated learning: What approach should you choose? In *Federated Learning: A Comprehensive Overview of Methods and Applications*, pages 281–312. Springer, 2022.

[CS10] Octavian Catrina and Amitabh Saxena. Secure computation with fixed-point numbers. In *FC*, pages 35–50. Springer, 2010.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 1976.

[DHNO19] Ivan Damgård, Helene Haagh, Michael Nielsen, and Claudio Orlandi. Commodity-based 2pc for arithmetic circuits. In *Cryptography and Coding*, pages 154–177. Springer, 2019.

[Dwo08] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*. ACM, 2015.

[FMM$^+$21] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, and Shaza Zeitouni. SAFELearn: Secure aggregation for private FL. In *SPW*, pages 56–62, 2021.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Symposium on Theory of computing*, pages 218–229. ACM, 1987.

[Gol04] Oded Goldreich. *Foundations of Cryptography*. Cambridge Un. Press, 1 edition, 2004.

[GWY$^+$18] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *CCS*, pages 619–633, 2018.

[Hei23] Henry Heitmann. DealSecAgg Experiments. https://github.com/henryheitmann/DealSecAgg/, 2023.

[HLS$^+$20] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[JNMALC22] Tayyebeh Jahani-Nezhad, Mohammad Ali Maddah-Ali, Songze Li, and Giuseppe Caire. Swiftagg: Communication-efficient and dropout-resistant secure aggregation for federated learning with worst-case security guarantees. In *ISIT*, pages 103–108. IEEE, 2022.

[JNMALC23] Tayyebeh Jahani-Nezhad, Mohammad Ali Maddah-Ali, Songze Li, and Giuseppe Caire. Swiftagg+: Achieving asymptotically optimal communication loads in secure aggregation for federated learning. *IEEE Journal Selected Areas in Communications*, 41(4):977–989, 2023.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, 2007.

[KMRR16] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv:1610.02527*, 2016.

[KRKR20] Swanand Kadhe, Nived Rajaraman, O Ozan Koyluoglu, and Kannan Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

[KS22] Marcel Keller and Ke Sun. Secure quantized training for deep learning. In *International Conference on Machine Learning*, pages 10912–10938. PMLR, 2022.

[KTC20] Youssef Khazbak, Tianxiang Tan, and Guohong Cao. MLGuard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning. In *ICCCN*, pages 1–9, 2020.

[LCW$^+$20] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *arXiv:2002.00211*, 2020.

[MBSP23] Julien Mirval, Luc Bouganim, and Iulian Sandu Popa. Federated learning on personal data management systems: Decentralized and reliable secure aggregation protocols. In *Proceedings of the 35th International Conference on Scientific and Statistical Database Management*, pages 1–12, 2023.

[MMR$^+$17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, pages 1273–1282. PMLR, 2017.

[MÖJC23] Mohamad Mansouri, Melek Önen, Wafa Ben Jaballah, and Mauro Conti. Sok: Secure aggregation based on cryptographic schemes for federated learning. *Proceedings on Privacy Enhancing Technologies*, 2023.

[MPP$^+$21] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[MTPBH21] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *PETS*, 2021.

[MWA$^+$23] Yiping Ma, Jess Woods, Sebastian Angel, Antigoni Polychroniadou, and Tal Rabin. Flamingo: Multi-round single-server secure aggregation with applications to private federated learning. *Cryptology ePrint Archive*, 2023.

[NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *SP*, pages 739–753, 2019.

[PKH22] Krishna Pillutla, Sham M. Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

[RG21] Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *arXiv:2007.07646 [cs]*, 2021.

[SGA21] Jinhyun So, Başak Güler, and A. Salman Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):479–489, 2021.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, 2004.

[SNY$^+$22] Jinhyun So, Corey J. Nolet, Chien-Sheng Yang, Songze Li, Qian Yu, Ramy E. Ali, Basak Guler, and Salman Avestimehr. LightSecAgg: a lightweight and versatile design for secure aggregation in federated learning. *Proceedings of Machine Learning and Systems*, 4:694–720, 2022.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *SP*, pages 3–18, 2017.

[SWDF23] Joshua Stock, Jens Wettlaufer, Daniel Demmler, and Hannes Federrath. Lessons learned: Defending against property inference attacks. In Sabrina De Capitani di Vimercati and Pierangela Samarati, editors, *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT)*, pages 312–323. SCITEPRESS, 2023.

[TLB$^+$21] Georgia Tsaloli, Bei Liang, Carlo Brunetta, Gustavo Banegas, and Aikaterini Mitrokotsa. DEVA: Decentralized, verifiable secure aggregation for privacy-preserving learning. In Joseph K. Liu, Sokratis Katsikas, Weizhi Meng, Willy Susilo, and Rolly Intan, editors, *Information Security*, pages 296–319. Springer International Publishing, 2021.

[TLG+19] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 14(6):2073–2089, 2019.

[WSZ+19] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM*, pages 2512–2520, 2019.

[Yao86] Andrew C. Yao. How to generate and exchange secrets. In *SFCS*, pages 162–167, 1986.

[ZLX+20] Chengliang Zhang, Suyi Li, J. Xia, W. Wang, F. Yan, and L. Yang. BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning. In *USENIX annual technical conference (USENIX ATC 20)*, pages 493–506, 2020.

7 Conclusion

ML algorithms have gained widespread popularity due to their remarkable ability to improve existing workflows and unlock new possibilities across various domains. Central to this success are modern training algorithms that automatically uncover useful patterns and rules from vast amounts of data. However, as this dissertation highlights, the pervasive use of data – often containing sensitive information – also introduces substantial privacy risks. As the reliance on ML models increases, safeguarding the privacy of data used for training these models becomes a critical challenge.

7.1 Summary

The focus of this dissertation has been on the protection of data used for training ML models. On one hand, training data properties can be extracted from trained ML models. In this area, white-box and black-box PIAs have been analyzed, discussed and further developed with contributions **C1** and **C2**. Contribution **C3** entails defense mechanisms against white-box and black-box PIAs, which have been thoroughly tested in experiments. On the other hand, when training data is not in one central data base but spread among multiple parties, decentralized training protocols such as FL enable a distributed training of ML models, with the decisive advantage of avoiding the transmission of training data. Contribution **C4** has investigated the performance, practical limits, and applicability of FL in different scenarios. Finally, contribution **C5** presents an FL protocol that provides additional anonymity for FL participants, such that their individual model updates cannot be linked to their identity by the coordinating party during or after training.

These contributions emerged while answering the four research questions posed in Chapter 1. Concluding my dissertation, this section revisits each research question and briefly summarizes the answers I could find in my work.

RQ1: How can training data properties be reconstructed from trained ML models?

In Chapter 3, PIAs were explored in a white-box scenario where an adversary has access to a trained target model's inner parameters and tries to reconstruct statistical properties of its training data. Specifically, the severity of the state-of-the-art white-box attack by Ganju et al. [GWY^{+}18] was demonstrated for three different data sets, with success rates as high as 99.3% or even 100% depending on the scenario. Experiments with the XAI tool LIME

– complemented with experiments in Appendix A – exhibited how ubiquitous property information is distributed in the weights of trained ANNs. This facilitates white-box PIAs, as an adversary has multiple opportunities to extract property information from trained model weights. Experiments with the visualization tool t-SNE additionally showed how deeply different properties in the training data set affect an ANN's weights: Without any context information, t-SNE could cluster apart most of the models by their training data property for two of three tested data sets.

In contrast, Chapter 4 also covered black-box PIAs, where the adversary does not have direct access to the model's internals but can only query a model and observe its output. To query the model and infer training data property distributions, the adversary uses an attack data set. A new approach to black-box PIAs was introduced, differing from related work in two ways: First, the proposal formulates PIAs as a regression problem rather than a classification task, which offers a more nuanced approach to understanding property distributions. Second, it allows adversaries to use an attack data set which is not necessarily part of the model's training data, broadening the scope for practical applications in real-world settings. In experiments with three data sets, the novel attack proved successful with R^2 test values between 0.63 and 0.72.

RQ2: Which strategies can help to mitigate the reconstruction of training data properties?

Recall that a successful defense strategy should meet three requirements, as mentioned in Section 1.2: It should be easy to implement, decrease the success rates of attackers and sustain model performance as best as possible. Adversarial learning, the state-of-the-art defense strategy against other attacks, such as model poisoning [GSS14, ACW18], is a good candidate to fulfill all three goals. For adversarial learning, the defender creates and utilizes their own adversary instance. This adversary is then used to steer the model away from revealing its secret, i.e., to intentionally lower the adversary's performance. In the case of defending against a PIA, this means that the model will be less prone to revealing a chosen sensitive training data property. Therefore, adversarial learning changes the training process but does not require changes in model architecture or training data, meeting the first requirement.

In Chapter 3, the adversarial learning strategy *property unlearning* is introduced for white-box PIA scenarios. It is designed as a post-hoc strategy, applied after model training has been completed. Experiments have shown that property unlearning works well when defending against a specific adversary instance, deterring the adversary from property inference while keeping the target model's utility high. However it fails to generalize, i.e., the defense is ineffective when the hardened model is attacked by another instance of a property inference adversary with the same goal. Therefore, this approach does not meet the second requirement stated in Section 1.2.

In contrast, the black-box defense strategy from Chapter 4 is applied *during* model training, with a parameter λ controlling the adversarial influence during the training process. Experiments have shown that $\lambda = 0.15$ is low enough to maintain the target model's performance (with an observed accuracy decrease of less than 0.15 percentage points) and high enough to prevent a successful attack, lowering the adversary's test R^2 to 0.07. Opposed to the white-box approach of Chapter 4, this strategy also generalizes well (see Appendix B), therefore meeting all three requirements for a successful defense strategy.

RQ3: What are the benefits and practical limits of FL?

In distributed ML training scenarios, FL has become a key technology. While sharing sensitive data is often not a viable option for companies and other organizations, FL enables training an ML model in a distributed way without the need to transfer any training data. As discussed in Chapter 5, FL therefore provides a basic level of training data privacy. The chapter also spotlights three simulated use cases, two of which focused on the performance of differently trained ML models. It was observed that the relative performance loss of models trained via FL was no greater than 7.2% compared to the respective models trained centrally on the same data.

On the other hand, the third use case of Chapter 5 showed how setting up an FL cooperation takes a lot of engineering effort. First, a framework suitable for all parties must be settled on. While the exchange of model updates could also be performed manually (e.g., by sending email attachments), the automated training orchestration by frameworks comes with huge manual labor savings in the long run. Second, the appropriate APIs and workflows must be implemented for all parties in order to ensure responsiveness during the training process. Third, the decentralized nature of the training data in FL scenarios can lead to a number of issues. Primarily it must be ensured that the training data is formatted and preprocessed in the same way for all parties. But even with unified data formats, hyperparameter optimization can be a particularly hard challenge, since the coordinating party – who is responsible for the model architecture and other training hyperparameters – usually does not have access to any of the actual training data. At least, this should be accounted for from the beginning by planning enough time for multiple iterations of HPO prior to the actual model training. This way, different settings can be tested before a final setting is decided on.

It should also be noted that the plain FL protocol [MMR+17] does not eliminate all risks regarding training data privacy. Hence, plain FL does not protect participants beyond the aforementioned basic training data privacy, as many publications on the topic have shown [LYY20]. One particular issue is that participants transfer their model updates to the central party in the clear, leading to the fourth research question.

RQ4: How can the anonymity of FL contributors be enhanced without sacrificing efficiency?

Chapter 6 contains a proposal for a cryptographic FL masking protocol called DealSecAgg. This protocol ensures the anonymity of FL participants by masking their updates prior to sending them to the aggregator. Masking is an established method in the area of secure aggregation for FL model updates, an active area of research. In the evaluation, DealSecAgg was compared to three similar protocols from related work. All of the other approaches have their individual shortcomings regarding computation efficiency. By introducing at least one independent dealer party who learns nothing besides the size of the trained model and the number of parties involved, runtimes of related work are reduced by up to 87.8% when using DealSecAgg. Additionally, DealSecAgg is efficiently handling dropouts, i.e., clients dropping out in the midst of a training process. The computation time of DealSecAgg is even marginally *decreased* when many clients drop out. This is a huge advantage compared to most related work, where the masks of dropouts need to be recovered through expensive cryptographic operations. Chapter 6 also contains a security proof, showing that DealSecAgg is provably secure against a semi-honest adversary unless the aggregator and all dealers are corrupted, or the aggregator and sufficiently many clients are corrupted.

While DealSecAgg outperforms related work in terms of runtimes, it requires at least one additional independent party, the dealer. Depending on the scenario the protocol is used in, this third party might be hard to find or instantiate. However, creating a dealer party should be feasible in most situations. For example, a non-profit organization could see the benefit of cooperating health institutions, or a trade association could help companies to train a model together. Given the speedup provided by using DealSecAgg, searching for such a dealer party would be a worthwhile investment.

7.2 Future Work

The research presented in this dissertation has addressed critical privacy concerns in machine learning, but many questions remain open for future exploration. Given the rapid pace of advancements in both privacy-preserving techniques and ML methodologies, there are numerous avenues for future research. In particular, several open challenges can build on the findings of this dissertation. This section combines a summary of future work discussed in previous chapters with new directions that emerged when writing up this dissertation.

Property Inference Attacks

In this dissertation, two adversarial learning approaches to mitigate PIAs were explored. Future research could include transferring both approaches to a *dynamic scenario*, in which the adversary may adapt their attack to the implemented defense strategy. For the presented white-box and black-box defense approaches, the most straightforward way forward would be to train new adversaries on defended shadow models and measure their success rates. For the black-box defense from Chapter 4, which is applied during training, also the adversary could be retrained *during* the training process, much like when training a GAN [GPAM+20]. Property unlearning, the white-box defense from Chapter 3, could similarly be applied during the training process. It would be interesting to see whether this – both the application during training and the dynamic setting – has positive effects on its generalizability. Note that the inspiration for the defense presented in Chapter 4, the adversarial approach to achieve ML fairness by Grari et al. [GRLD20], also makes use of a dynamic adversary. However their adversary has another goal and is much more simple, since it does not need anything beyond the training data and the current output of the target model (in contrast to the PIA adversary trained on the output of multiple shadow models).

Another direction for future work could be to find alternative approaches for both white-box and black-box PIA defenses. A candidate strategy could be *knowledge distillation (KD)*. The idea of KD is to lower the memory capacity of an ANN. The primary goal of this is to reduce the size of the model, thereby reducing storage and computation requirements for its use [Hin15]. The first step of KD is to train a (potentially large) model and store its detailed probability outputs for the training data. To train the distilled, smaller network, the original class labels of the training set (*hard class labels*) are replaced with the trained model's probability output (*soft labels*). When training the distilled model, it can therefore make use of the large model's training encoded in the preprocessed soft labels. Other works have shown how this technique can make a model more robust towards ML attacks, e.g., against adversarial examples [PMW+16] and MIAs [SH21]. However for other attacks such as model inversion and model stealing, another publication could not observe a significant decrease in attack success rates by using KD [LWH+22]. It would be interesting to evaluate the impact of KD on PIA success rates and the associated decrease in model performance.

Another way to expand the research of this dissertation is to explore the applicability of the developed defense strategies of Chapters 3 and 4 on other PIA attacks. One example for this would be the novel black-box KL divergence attack by Suri et al. [SLCE23], as introduced in Section 2.4.3. Since the black-box defense approach from Chapter 4 is generic and any adversary could be plugged into the algorithm, the technical barriers for its implementation are low. Experiments on defending against the KL divergence attack could determine whether the findings of this work generalize to other attacks. Particularly, the results could show whether the observed privacy-utility tradeoff in Chapter 4 can be reproduced in other attack scenarios.

As far as new types of PIA strategies are concerned, an open challenge is to further reduce the effort required for an attack. Suri et al. have shown how the KL divergence attack succeeds with training less shadow models than comparable black-box attacks. However for other attacks such as MIAs, there are approaches without training any shadow models at all [LZ21]. This approach is not directly transferrable to the property inference setting, since MIAs have a different goal than PIAs. But if there were certain observations to be made about models and their properties which can lead to a PIA design without any auxiliary data sets and shadow models, this would significantly advance the PIAs research community.

Masking for Federated Learning

Regarding FL, a secure aggregation masking scheme called DealSecAgg was proposed in Chapter 6. Note that DealSecAgg was designed for the FedAvg aggregation function, which means that the aggregator computes the average of all the clients' model updates at the end of each training round. Computationally, this is the most simple aggregation function, since it only involves summing up the client updates and dividing by the number of contributors afterwards. Extending DealSecAgg to work for other aggregation functions as well would be a major feature to broaden the applicability of the protocol. Lately *byzantine-robust aggregation functions* have been proposed, in order to account for single malicious contributions which may easily skew the average model update of FedAvg[1]. The term *byzantine-robust* means that these alternatives to FedAvg are less sensitive to such outliers [FCJG20]. As an example, one could simply calculate the median update instead of the mean, since outliers have less impact on the median. However this is not trivial to implement for the DealSecAgg protocol, which relies on forming sums of both masked model updates and masks. The same applies to *Krum* [BEMGS17], another popular byzantine-robust aggregation function [FCJG20]. Krum calculates pairwise Euclidean distances between all client updates. For n clients, including at most c corrupted clients, the distances between each contributed model update and its closest $m-c-2$ neighboring model updates (with respect to the Euclidean distance) are calculated. The model with the smallest sum of respective distances is then selected as the global model for the

1. The term *byzantine* means that clients might behave arbitrarily, i.e., contribute model updates with unexpected values [BEMGS17].

next round. Since the aggregator has to analyze individual model updates for Krum and masking approaches prevent the aggregator from having access to such individual contributions, this is not straightforward to implement either. DealSecAgg could however be adapted such that the dealer splits the n clients into a constant number of equally sized client groups $g \in G$, say $|G| = 10$ with $\sum_{g \in G} |g| = n$. As long as there are enough clients taking part in the training process, masks and model updates could be aggregated within each group. The idea is that if the number of byzantine clients c is small (say $c < |G|$), the updates of byzantine clients could be isolated such that there are groups without byzantine updates. Inspired by Krum, a cautious aggregator might then update the global model with the average update of the group with the smallest sum of distances, when compared to the other groups' average updates. Since splitting clients into groups could harm their anonymity, the number of groups and the minimal number of participating clients must be carefully chosen before applying such a scheme in practice.

Beyond this, future work could include transforming DealSecAgg into a setting with malicious parties, where either a corrupted aggregator or corrupted clients may deviate from the protocol. This could be achieved by introducing consistency checks, as discussed in Section 6.7.1.

Privacy and Security for Machine Learning

The landscape of privacy and security in ML is rapidly evolving, particularly with the increasing deployment of large-scale models like LLMs. As the demand for ML continues to grow, so does the importance of safeguarding sensitive data throughout the training process. The future of FL and other ML systems could lie in the integration of advanced cryptographic PPML methods such as HE and SMPC. With the continuous advancements in computational power and dedicated hardware [ZCY⁺24], these privacy-preserving tools could become increasingly practical and widely adopted, allowing for the secure and private deployment of ML models in a variety of sensitive applications.

Complementing ongoing efforts to understand and mitigate different forms of privacy leakage in ML applications, a more widespread use of such PPML techniques could play a significant role to make the future of ML more privacy friendly and secure.

7.3 Final Outlook

The advent of ML algorithms has transformed numerous industries, offering unprecedented opportunities for innovation and efficiency. However, as these technologies become increasingly embedded in our daily lives, the need for robust privacy protection is more critical than ever. This dissertation has focused on addressing the privacy concerns associated with ML systems, particularly in the context of PIAs and FL.

By developing novel defense strategies for PIAs and proposing advanced protocols to ensure the anonymity of FL participants, this work has contributed to the growing body of research focused on safeguarding sensitive data in the age of machine learning. While significant progress has been made, it is clear that the journey is far from complete. As ML continues to evolve, so too must our strategies for mitigating privacy risks.

Looking ahead, the challenges of ensuring data privacy will only become more complex, as new models and techniques emerge. The growing scale of ML models, the rise of large-scale deployments such as generative models, and the increasing interconnectivity of systems all pose new risks. It is essential that the research community continues to explore innovative solutions to prevent privacy breaches and develop secure frameworks that allow ML models to thrive while respecting individuals' privacy.

This dissertation has laid a foundation, but the need for further research remains. As we move forward, we must refine existing methodologies, expand our understanding of emerging attack vectors, and continue pushing the boundaries of PPML. The future of ML is not only about making models smarter but also about making them safer and more privacy friendly. The work presented here serves as one step in this ongoing effort, and the path ahead promises to be both challenging and rewarding.

References

[ACW18] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[ALML16] Julia Angwin, Jeff Larson, Surya Mattu, and Kirchner Lauren. Machine bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, 2016. Accessed on December 17, 2024.

[AMJ18] David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

[AMS$^+$15] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[BBG$^+$20] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

[BCH22] Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1138–1156. IEEE, 2022.

[BCL23] Christopher Burger, Lingwei Chen, and Thai Le. Are your explanations reliable? Investigating the stability of LIME in explaining text classifiers by marrying XAI and adversarial attack. *arXiv preprint arXiv:2305.12351*, 2023.

[BCMC19] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International conference on machine learning*, pages 634–643. PMLR, 2019.

[BEMGS17] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

[BIK$^+$17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, pages 1175–1191. ACM, 2017.

[CHN+23] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270, 2023.

[CSR+20] Rosario Cammarota, Matthias Schunter, Anand Rajan, Fabian Boemer, Ágnes Kiss, Amos Treiber, Christian Weinert, Thomas Schneider, Emmanuel Stapf, Ahmad-Reza Sadeghi, et al. Trustworthy AI inference systems: An industry research view. *arXiv preprint arXiv:2008.04449*, 2020.

[CTW+21] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[CYZF20] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 343–362, 2020.

[DBC+19] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Adebayo Olusola Adetunmbi, Opeyemi Emmanuel Ajibuwa, et al. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6), 2019.

[DDO+20] Timo M Deist, Frank JWM Dankers, Priyanka Ojha, M Scott Marshall, Tomas Janssen, Corinne Faivre-Finn, Carlotta Masciocchi, Vincenzo Valentini, Jiazhou Wang, Jiayan Chen, et al. Distributed learning on 20 000+ lung cancer patients–the personal health train. *Radiotherapy and Oncology*, 144:189–200, 2020.

[Dwo06] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[FCJG20] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[FH19] Matthias Feurer and Frank Hutter. Hyperparameter optimization. *Automated machine learning: Methods, systems, challenges*, pages 3–33, 2019.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In

Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pages 1322–1333, 2015.

[Goo24] Google. A guide to Google Search ranking systems. https://developers.google.com/search/docs/appearance/ranking-systems-guide, 2024. Accessed on October 17, 2024.

[GPAM$^+$14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[GPAM$^+$20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[GRLD20] Vincent Grari, Boris Ruf, Sylvain Lamprier, and Marcin Detyniecki. Achieving fairness with decision trees: An adversarial approach. *Data Science and Engineering*, 5(2):99–110, 2020.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[GWY$^+$18] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[HDJ19] István Hegedűs, Gábor Danner, and Márk Jelasity. Gossip learning as a decentralized alternative to federated learning. In José Pereira and Laura Ricci, editors, *Distributed Applications and Interoperable Systems*, pages 74–90, Cham, 2019. Springer International Publishing.

[Hei23] Henry Heitmann. DealSecAgg Experiments. https://github.com/henryheitmann/DealSecAgg/, 2023.

[HHB19] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. Monte carlo and reconstruction membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019.

[Hin15] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[HKR$^+$18] Andrew Hard, Chloé M Kiddon, Daniel Ramage, Francoise Beaufays, Hubert Eichner, Kanishka Rao, Rajiv Mathews, and Sean Augenstein. Federated learning for mobile keyboard prediction. https://arxiv.org/abs/1811.03604, 2018.

[HMDDC17] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. *arXiv preprint arXiv:1705.07663*, 2017.

[HMP$^+$23] Valentin Hartmann, Léo Meynent, Maxime Peyrard, Dimitrios Dimitriadis, Shruti Tople, and Robert West. Distribution inference risks: Identifying and mitigating sources of leakage. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 136–149. IEEE, 2023.

[HN20] Hannes Hapke and Catherine Nelson. *Building machine learning pipelines*. O'Reilly Media, 2020.

[Hu23] Krystal Hu. ChatGPT sets record for fastest-growing user base - analyst note. https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/, 2023. Accessed on October 17, 2024.

[HZ24] Qianyu Huang and Tongfang Zhao. Data collection and labeling techniques for machine learning. *arXiv preprint arXiv:2407.12793*, 2024.

[JSMA19] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019.

[KB18] Veton Kepuska and Gamal Bohouta. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *8th annual computing and communication workshop and conference (CCWC)*, pages 99–103. IEEE, 2018.

[KMRR16] Jakub Konečný, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[KPS$^+$24] Moritz Kirschte, Thorsten Peinemann, Joshua Stock, Carlos Cotrini, and Esfandiar Mohammadi. S-GBDT: Frugal differentially private gradient boosting decision trees. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2024.

[KRKR20] Swanand Kadhe, Nived Rajaraman, O Ozan Koyluoglu, and Kannan Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *IEEE*, 1998.

[LHS+24] William Lotter, Michael J Hassett, Nikolaus Schultz, Kenneth L Kehl, Eliezer M Van Allen, and Ethan Cerami. Artificial intelligence in oncology: Current landscape, challenges, and future directions. *Cancer Discovery*, 14(5):711–726, 2024.

[LWH+22] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. {ML-Doctor}: Holistic risk assessment of inference attacks against machine learning models. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4525–4542, 2022.

[LYY20] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

[LZ21] Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 880–895, 2021.

[MBG+23] Johanna Ansohn McDougall, Alessandro Brighente, Willi Großmann, Ben Ansohn McDougall, Joshua Stock, and Hannes Federrath. Love is in the air - location verification of ADS-B signals using distributed public sensors. In *IEEE International Conference on Communications (ICC)*, pages 6040–6045. IEEE, 2023.

[MGC22] Saeed Mahloujifar, Esha Ghosh, and Melissa Chase. Property inference from poisoning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1120–1137. IEEE, 2022.

[MMR+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[MMS+21] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.

[MÖJC23] Mohamad Mansouri, Melek Önen, Wafa Ben Jaballah, and Mauro Conti. Sok: Secure aggregation based on cryptographic schemes for federated learning. *Proceedings on Privacy Enhancing Technologies*, 2023.

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[NCC+20] Jianjun Ni, Yinan Chen, Yan Chen, Jinxiu Zhu, Deena Ali, and Weidong Cao. A survey on theories and applications for self-driving cars based on deep learning methods. *Applied Sciences*, 10(8):2749, 2020.

[NSH18a] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, volume 2018, pages 1–15, 2018.

[NSH18b] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *CCS*, 2018.

[OHJ13] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[PAC18] Ivens Portugal, Paulo Alencar, and Donald Cowan. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205–227, 2018.

[PMW+16] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[RG23] Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *ACM Computing Surveys*, 56(4):1–34, 2023.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should I trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[SE22] Anshuman Suri and David Evans. Formalizing and estimating distribution inference risks. *Proceedings on Privacy Enhancing Technologies*, 2022.

[SH21] Virat Shejwalkar and Amir Houmansadr. Membership privacy for machine learning models through knowledge transfer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9549–9557, 2021.

[SHSD24] Joshua Stock, Henry Heitmann, Janik Noel Schug, and Daniel Demmler. Dealsecagg: Efficient dealer-assisted secure aggregation for federated learning. In *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES)*, 2024.

[SHWF23] Joshua Stock, Oliver Hauke, Julius Weißmann, and Hannes Federrath. The applicability of federated learning to official statistics. In *International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, pages 70–81. Springer, 2023.

[SLCE23] Anshuman Suri, Yifu Lu, Yanjin Chen, and David Evans. Dissecting distribution inference. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 150–164. IEEE, 2023.

[SLRF24] Joshua Stock, Lucas Lange, Erhard Rahm, and Hannes Federrath. Property inference as a regression problem: Attacks and defense. In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT)*, pages 876–885. SciTePress, 2024.

[SM21] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *USENIX Security*, 2021.

[SNY$^+$22] Jinhyun So, Corey J. Nolet, Chien-Sheng Yang, Songze Li, Qian Yu, Ramy E. Ali, Basak Guler, and Salman Avestimehr. LightSecAgg: a lightweight and versatile design for secure aggregation in federated learning. *Proceedings of Machine Learning and Systems*, 4:694–720, 2022.

[SPB$^+$22a] Joshua Stock, Tom Petersen, Christian-Alexander Behrendt, Hannes Federrath, and Thea Kreutzburg. Privatsphärefreundliches maschinelles lernen: Teil 1: Grundlagen und verfahren. *Informatik Spektrum*, 45(2):70–79, 2022.

[SPB$^+$22b] Joshua Stock, Tom Petersen, Christian-Alexander Behrendt, Hannes Federrath, and Thea Kreutzburg. Privatsphärefreundliches maschinelles lernen: Teil 2: Privatsphäreangriffe und privacy-preserving machine learning. *Informatik Spektrum*, 45(3):137–145, 2022.

[SRS17] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, pages 587–601, 2017.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[Sto23] Joshua Stock. Federated learning simulations. https://github.com/joshua-stock/fl-official-statistics, 2023.

[Sto25a] Joshua Stock. Property unlearning experiments. https://github.com/joshua-stock/property-unlearning, 2025.

[Sto25b] Joshua Stock. Regression property inference experiments. https://github.com/joshua-stock/regression-property-inference, 2025.

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[SWDF23] Joshua Stock, Jens Wettlaufer, Daniel Demmler, and Hannes Federrath. Lessons learned: Defending against property inference attacks. In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT)*, pages 312–323. SciTePress, 2023.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[TMS+21] Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. Mitigating membership inference attacks by self-distillation through a novel ensemble architecture. *USENIX Sec.*, 2021.

[TSN20] Shruti Tople, Amit Sharma, and Aditya Nori. Alleviating privacy attacks via causal learning. In *International Conference on Machine Learning*, pages 9537–9547. PMLR, 2020.

[Vap91] Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

[VR24] Rohit S Vilhekar and Alka Rawekar. Artificial intelligence in genetics. *Cureus*, 16(1), 2024.

[WAWR20] Alexander Warnecke, Daniel Arp, Christian Wressnegger, and Konrad Rieck. Evaluating explanation methods for deep learning in security. In *2020 IEEE european symposium on security and privacy (EuroS&P)*, pages 158–174. IEEE, 2020.

[WHSS+21] Stefanie Warnat-Herresthal, Hartmut Schultze, Krishnaprasad Lingadahalli Shastry, Sathyanarayanan Manamohan, Saikat Mukherjee, Vishesh Garg, Ravi Sarveswara, Kristian Händler, Peter Pickkers, N Ahmad Aziz, et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature*, 594(7862):265–270, 2021.

[WWC$^+$19] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. Evaluating word embedding models: Methods and experimental results. *APSIPA transactions on signal and information processing*, 8:e19, 2019.

[YGFJ18] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018.

[ZCSZ21] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. Property inference attacks against gans. *arXiv preprint arXiv:2111.07608*, 2021.

[ZCY$^+$24] Junxue Zhang, Xiaodian Cheng, Liu Yang, Jinbin Hu, Ximeng Liu, and Kai Chen. Sok: Fully homomorphic encryption accelerators. *ACM Computing Surveys*, 56(12):1–32, 2024.

[Zho21] Zhi-Hua Zhou. *Machine learning*. Springer nature, 2021.

[ZTO21] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of dataset properties in Multi-Party machine learning. In *30th USENIX security symposium (USENIX Security 21)*, pages 2687–2704, 2021.

A Revisiting the Explainability of White-Box Property Inference

The focus of the experiments in Chapter 3 is on white-box PIAs. As discussed, the state-of-the-art approach for constructing such adversaries to attack fully-connected ANNs is based on permutation invariance [GWY$^+$18]. Since the explainability experiment in Section 3.6 does not fully account for this permutation invariance, additional experimental evidence is provided here.

For the following two experiments, the same experimental setup as in Section 3.6 is used: Two sets of shadow models are trained on the MNIST data set for handwritten digit recognition [LBBH98]. On each set of shadow models, one PIA adversary is trained. Both adversaries then attack the same target model.

As in Section 3.6, the tool local interpretable model-agnostic explanations (LIME) [RSG16] is used in the first experiment – but its results are evaluated in a different way. In the second experiment, integrated gradients (IG) [STY17] is applied. The additional use of IG is to make sure that the XAI results are valid, motivated by shortcomings of the LIME approach.

Limitations of LIME

In an evaluation of XAI tools by Warnecke et al. [WAWR20], IG outperforms LIME in 4 out of 6 evaluation criteria, while the remaining two categories show a similar performance of the two tools. When explaining the decisions of PIA adversaries, the XAI tools are employed in a very limited scope: one fixed input for two classifiers with the same task. Therefore, many robustness and security concerns about LIME [AMJ18, BCL23, WAWR20] do not apply to the use case at hand.

However, Warnecke et al. raise issues which could limit the significance of the LIME experimental results. One of them is captured by the criterion *descriptive accuracy*: When the most important attributes of a sample x leading to a classifier's decision are removed from x, the accuracy of the classifier should decrease [WAWR20, Def. 2]. The other "general criterion" is *descriptive sparsity*, measuring how well an XAI tool assigns important weights to input attributes – understanding sparsity in the sense that only a few attributes should be deemed important in a good explanation [WAWR20, Def. 3]. Out of the six tested XAI tools in the paper, IG has the best performance regarding both general criteria, outperforming LIME and the other four tools. This indicates that the quality of IG explanations tends to be inherently higher than the quality of LIME explanations. Furthermore, LIME treats the explained model as a black box, while IG is a white-box method calculating its explanations directly from the weights of the classifiers which are explained. Since access to the internal weights of the PIA meta-classifier is not an issue in the experiments at hand, the LIME experiment can be repeated with the stronger white-box XAI tool IG.

Experimental Setup

Recall that within the experiment in Section 3.6, two PIA adversaries have been trained with the same goal: inferring a specific property of a target model's training data. When these two adversaries are attacking the same target model M, the explainability tool LIME is used to highlight the parts in the target model's weights which are important for the decision of both adversaries. To quantify importance, LIME assigns importance values to each input value – which correspond to the target model parameters in the case at hand.

In order to account for the permutation invariance of neurons within a layer of a fully connected ANN, only the distributions of these importance values for the weights of M are considered in the experiments here[1], beginning with the neuron weights of the first layer.

Results

Table A.1 describes the distributions of the experiment outcomes: For both XAI tools LIME and IG, the distributions of the explanations, i.e., importance vectors, calculated for both adversary instances 1 and 2 are compared. To provide better comparability, the importance vectors are also scaled to a range between 0 and 1 and display both variants (scaled and unscaled) in the table.

Apart from similar metrics in the scaled LIME explanations, the distributions of importance vectors are very different, especially those calculated by IG. This is also reflected in Figure A.1, where the histograms for both adversary instance explanations are plotted in one graph for each scenario (LIME and IG, scaled and unscaled). It is apparent that the distributions differ in each scenario.

1. The code for the experiments of this Appendix is available at [Sto25a], specifically in the files `mnist/LIME-revisited.ipynb` and `mnist/ig-explainability.ipynb`.

Tool	LIME				IG			
Scaled	no		yes		no		yes	
PIA inst.	1	2	1	2	1	2	1	2
Mean	7.77×10^{-6}	9.82×10^{-7}	0.537	0.488	-3.41×10^{-9}	8.33×10^{-7}	0.613	0.400
Std Dev	8.98×10^{-5}	6.21×10^{-5}	0.105	0.119	1.67×10^{-6}	1.21×10^{-5}	0.032	0.070
Min	-4.54×10^{-4}	-2.54×10^{-4}	0.000	0.000	-2.03×10^{-5}	-6.77×10^{-5}	0.000	0.000
25% Q.	-5.23×10^{-5}	-4.09×10^{-5}	0.467	0.408	-3.35×10^{-7}	-4.95×10^{-7}	0.603	0.366
50% Q.	7.95×10^{-6}	9.01×10^{-6}	0.538	0.488	-1.20×10^{-9}	1.75×10^{-7}	0.614	0.396
75% Q.	6.83×10^{-5}	4.27×10^{-5}	0.608	0.568	3.32×10^{-7}	6.29×10^{-6}	0.624	0.432
Max	4.05×10^{-4}	2.69×10^{-4}	1.000	1.000	1.28×10^{-5}	1.04×10^{-4}	1.000	1.000
Range	8.59×10^{-4}	5.23×10^{-4}	1.000	1.000	3.30×10^{-5}	1.71×10^{-4}	1.000	1.000
Variance	8.06×10^{-9}	3.85×10^{-9}	0.011	0.014	1.14×10^{-12}	1.48×10^{-10}	0.001	0.005

Table A.1: Descriptive statistics of the attribute importance values produced by the two XAI tools LIME and IG. inst.=instance, Q.=Quantile.

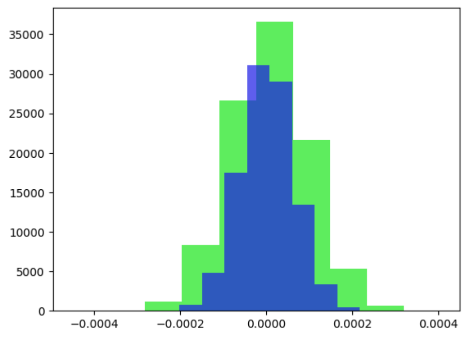
Discussion and Conclusion

The main conjecture of Section 3.6 can be summarized as follows:

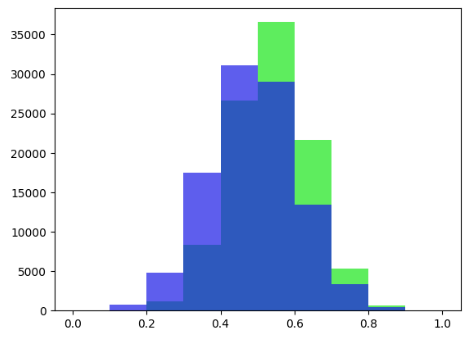
Conjecture 1. *Two white-box PIA adversary instances with the same goal rely on different parts of a target model M to infer a training data property from M.*

If this conjecture were not true, the explanation of an XAI tool would be expected to reveal that both instances rely on the same weights of M's trained weights, i.e., assign similar importance values to the neuron weights of one layer within M. However, the experiments show that both LIME and IG assign different importance values to the weights of neurons within M's first layer[2]. Hence, further experimental evidence supporting Conjecture 1 has been gathered, such that the same conclusion as in Section 3.6 can be drawn, i.e., the conjecture seems to be true.

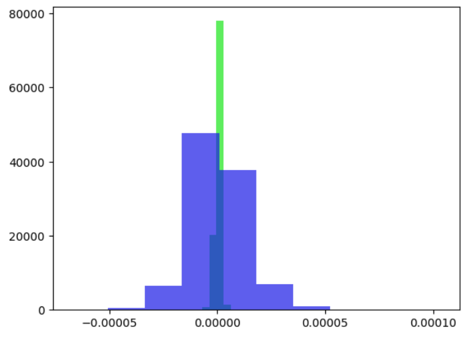
2. The results for the other layers are omitted, since the differences for the first layer suffice for the validation of Conjecture 1.



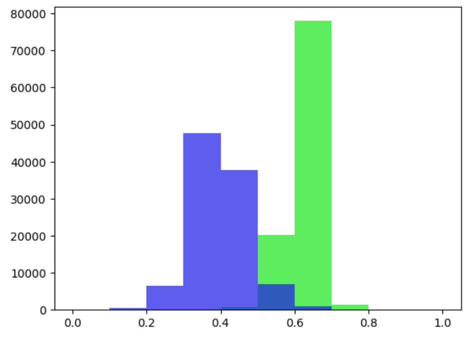
(a) Unscaled LIME.



(b) Scaled LIME.



(c) Unscaled IG.



(d) Scaled IG.

Figure A.1: Histograms of importance values when two PIA adversary instances (green and blue histogram) classify the same target model *M*. The importance values from both tools LIME and IG are each shown as they are calculated by the tools (unscaled) and scaled from 0 to 1.

B Generalizability Experiment for the Black-Box Property Inference Defense

In Chapter 4, a strategy to defend against black-box PIAs is introduced and evaluated. The goal of the defense, as explained in Section 1.2, is to harden target models against the inference of a specific training data property. This Appendix is an extension of the experiments of Section 4.5.3. Specifically, the experiments presented here[1] are designed to gather evidence for answering the following question, related to the second goal of **RQ2** in Section 1.2:

Question 1. *Consider two adversaries $\mathscr{A}$ and $\mathscr{A}_0$, both aimed at inferring a property p from target models $m \in M$ but trained on the outputs of different shadow models. When using $\mathscr{A}$ during adversarial learning of model m as introduced as a defense strategy in Section 4.4, how well can another adversary $\mathscr{A}_0$ infer p from m?*

Recall that the *white-box* PIA defense of Chapter 3 did not meet this generalizability goal. In contrast, this Appendix is dedicated to the *black-box* regression setting.

Experimental Setup

The same initial experimental setup as in Section 4.5.3 is used: An adversary $\mathscr{A}$ is trained on the output of shadow models with different training data property values $p_v \in \{0.1, 0.2, \ldots, 0.9\}$. For the UTKFace data set, 200 shadow models are trained on auxiliary data sets with the respective p_v, while 400 shadow models per property value p_v are trained and used for CIFAR-10. After successfully training the adversary $\mathscr{A}$, it is used for adversarial learning to harden five target models $m \in M$ per property value p_v. The training data of the target models $m \in M$ is disjunct from the training data of shadow models used for adversary training. For this experiment, 5 additional adversaries $\mathscr{A}_0, \ldots, \mathscr{A}_4$ are trained with the same target property as $\mathscr{A}$. The additional adversaries are each trained on different sets of shadow models, which are disjunct from the set of shadow models used during $\mathscr{A}$'s training.

1. The code for the experiments of this Appendix is available at https://github.com/joshua-stock/regression-property-inference/blob/main/src/generalizability.ipynb

	no defense	defended models
	$\lambda = 0.0$	$\lambda = 0.15$
UTKFace	0.58 (0.63)	0.30 (0.07)
CIFAR-10	0.64 (0.64)	0.32 (0.07)

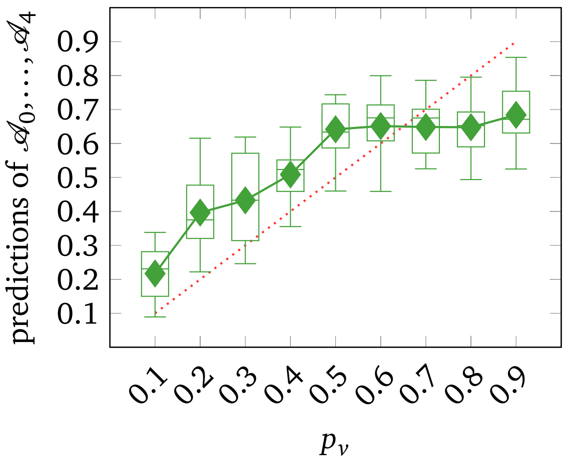
Table B.1: The performance of the black-box defense as R^2 values when *other* adversaries infer the property from defended models. Values in brackets are the reference performance of the adversary used during adversarial learning (as presented in Table 4.2).

Hence, there are five target models for each of the nine property values p_v. These models are attacked by five different adversaries, yielding $5*9*5 = 255$ measurements per setting. The training of target models is conducted twice, once without adversarial learning ($\lambda = 0.0$) and once with an adversarial influence of $\lambda = 0.15$.

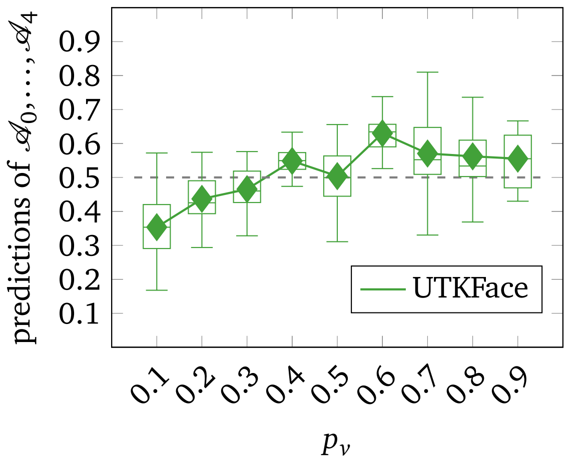
Results

The results of the experiment are summarized in Table B.1: The mean test R^2 performance of the additional adversaries $\mathscr{A}_0, \dots, \mathscr{A}_4$ on undefended target models is compared to their performance on defended models with $\lambda = 0.15$. For the UTKFace data set, the R^2 drops from 0.58 to 0.30, while the CIFAR-10 adversaries exhibit a similar decrease from 0.64 to 0.32. This is compared to the performance of adversary $\mathscr{A}$, which was used during the adversarial learning of the target models: Here, the R^2 value drops from 0.64 (resp. 0.63) to 0.07 for both data sets.

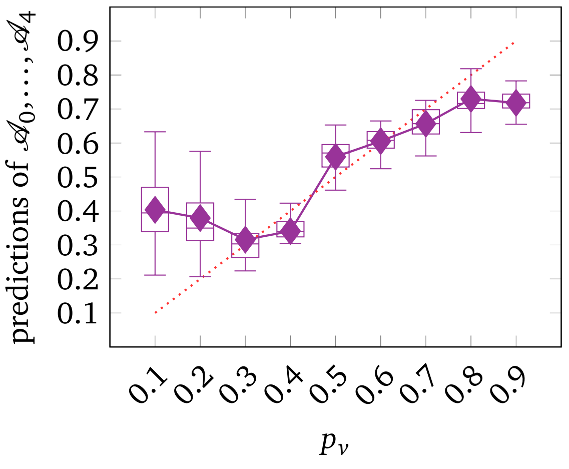
Figure B.1 offers more details on the performance of adversaries $\mathscr{A}_0, \dots, \mathscr{A}_4$: For both data sets UTKFace and CIFAR-10 and both settings $\lambda \in \{0.0, 0.15\}$, the adversary predictions are plotted on the y-axis, for each of the property values p_v on the x-axis. The dotted red lines represent a perfect adversary prediction, while the dashed gray lines display the target of the defense, i.e., a constant adversarial output of 0.5. The performance in the four different settings as described in Table B.1 is reflected in these plots, i.e., the adversary outputs for defended models is generally closer to 0.5. The output values of the adversaries are not significantly more spread out in one scenario or the other: The range of values, displayed by the space between the “whiskers” in each boxplot, is similar in the four graphs.



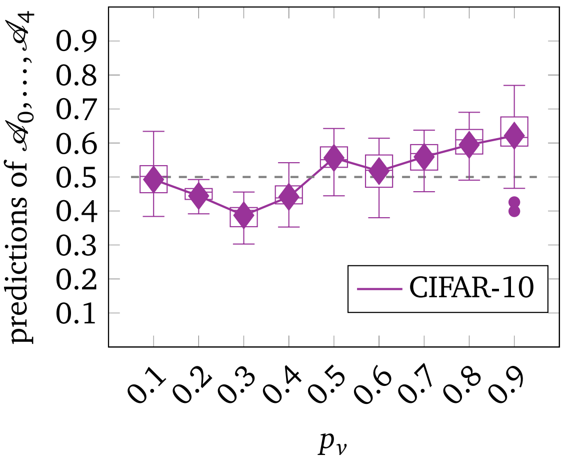
(a) Undefended UTKFace models ($\lambda = 0.0$)



(b) Defended UTKFace models ($\lambda = 0.15$)



(c) Undefended CIFAR-10 models ($\lambda = 0.0$)



(d) Defended CIFAR-10 models ($\lambda = 0.15$)

Figure B.1: Plotted results of the generalizability experiment: Results for the UTKFace data set are displayed in the top row, results for CIFAR-10 are in the bottom row. The plots in the left column display the output of adversaries $\mathscr{A}_1, \dots, \mathscr{A}_4$ for undefended models, the right column shows the outputs for defended models with $\lambda = 0.15$.

Discussion and Conclusion

The defense strategy of Chapter 4 works very well for defending against the adversary which was used during the adversarial learning of a target model (reflected by the reference values of Table B.1). While the defense mechanism performs slightly worse to protect against other adversaries, such as $\mathscr{A}_0, \ldots, \mathscr{A}_4$ in the experiment above, it still achieves a major disruption in the adversaries' performance. On one hand, this is reflected in significantly worse R^2 values for the adversaries, dropping from ~ 0.6 for undefended models to ~ 0.3 when the defense was applied. On the other hand, the performance decrease is also evident in the plots of Figure B.1, where the mean outputs for most of the property values p_v get worse, i.e., the difference between the predicted values and the true values is generally increased by the defense strategy.

Note that, as discussed in Section 4.6, setting the target property value p_v for all target models to 0.5 is an artificial setting for a comprehensive experimental setup. In real applications, it might be wise to deliberately choose a target p_v which is far away from the real p_v, or a random value between 0 and 1.

In conclusion, the black-box PIA defense strategy of Chapter 4 does not only protect against the adversarial instance $\mathscr{A}$ which was used during adversarial learning, but also interferes with the property inference of other adversaries with the same goal. Considering the results of the experiment above, an answer to Question 1 can be formulated as follows: Compared to the effect of the defense on $\mathscr{A}$, the effect on other adversaries such as $\mathscr{A}_0$ is smaller. Still, the defense achieves significant performance disruptions on the other adversaries, i.e., generalization could be observed in the experiments.

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. Sofern im Zuge der Erstellung der vorliegenden Dissertationsschrift generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

Lüneburg, 30. März 2025

Joshua Stock