Secure and Private Machine Learning

Dissertation zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt am

Fachbereich Mathematik und Informatik
der Freien Universität Berlin

von

Franziska Boenisch

Berlin, 2022

Erstgutachter:	Prof. Dr. Marian Margraf
Zweitgutachter:	Prof. Dr. Nicolas Papernot
Tag der Disputation:	17.11.2022

Declaration of authorship

Name: Boenisch
First Name: Franziska

I declare to the Freie Universität Berlin that I have completed the submitted dissertation independently and without the use of sources and aids other than those indicated. The present thesis is free of plagiarism. I have marked as such all statements that are taken literally or in content from other writings. This dissertation has not been submitted in the same or similar form in any previous doctoral procedure.

I agree to have my thesis examined by a plagiarism examination software.

Berlin, August 9th, 2022

_____
Franziska Boenisch

Abstract

In recent years, the advances of Machine Learning (ML) have led to its increased application within critical applications and on highly sensitive data. This drew attention to the aspects of ML security and privacy. ML models should operate correctly and not reveal sensitive data that they were trained on. However, assessing and implementing ML security and privacy is a challenging task. This is, first of all, because the effects of current ML practices on these aspects are not yet fully understood. Consequently, the array of known risks still contains a multitude of blind spots. In a similar vein, the implicit assumptions under which ML security and privacy can be achieved in a given practical application often remain unexplored.

In this work, we present a study on security and privacy in ML that contributes to overcoming the existing limitations. Therefore, we first provide insights into the current state of security and privacy of ML in practice by surveying ML practitioners. We find that ML practitioners exhibit a particularly low awareness when it comes to ML privacy and that they trust third-party frameworks and services for its implementation. These insights motivate the necessity to investigate ML privacy more in depth. We do so with a focus on Federated Learning (FL) since FL is a commonly used framework for real-world applications that affect hundreds of thousands of users and their private data. In this setup, we study privacy leakage from ML models and show that model gradients can directly leak private information on large fractions of their sensitive training data. Building on these findings, we extend existing research on maliciously attacking the privacy of this training data by proposing a novel attack vector, namely adversarial initialization of the model weights. By thoroughly exploring this attack vector, we assess the assumptions on trust required to obtain meaningful privacy guarantees in FL. In particular, we focus on trust assumptions regarding the central server in FL. Finally, to explore the intersection of ML security and privacy, we investigate what impact the implementation of privacy guarantees has on ML models' robustness. Eventually, through this work, we aim to advocate the importance of a secure and privacy-preserving design of ML methods—in particular when these are applied in real-world scenarios.

Acknowledgements

First of all, I would like to thank my doctoral advisor Prof. Dr. Marian Margraf for supporting me throughout the time of my dissertation in all of my endeavors. His door was always open so I could turn to him with questions and for advice on all kinds of situations.

I also would like to express gratitude to my co-advisor Prof. Dr. Nicolas Papernot for his guidance, feedback, insightful comments, and support in my research, especially during my internships at the University of Toronto and at the Vector Institute. His views and visions inspired me to continue working on research after my PhD. He taught me how to be an ethical researcher and I admire him for his remarkable skills as a researcher, professor, mentor, and group leader.

Furthermore, thank you to my wonderful colleagues both at the Vector Institute in Toronto and the Fraunhofer AISEC in Berlin for their great support in all facets. It was inspiring to talk to them about research, to develop and execute new ideas together, and to share the experience of growing as a researcher and as a person over the years. In particular, I want to thank Adam for his feedback on my dissertation. I am also very thankful for all the students I could supervise during the time of my dissertation. I enjoyed our exchange about machine learning, privacy and security, and research as a whole.

Finally, I would like to thank my parents for giving me life, education, and their unlimited support, both academically and emotionally, in every situation. I am very grateful to have them in my life.

Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
BA_2	Boundary Attack
CCPA	California's Consumer Privacy Act
CDP	Centralized Differential Privacy
CNN	Convolutional Neural Network
CW_2	Carlini and Wagner
DDP	Distributed Differential Privacy
DP	Differential Privacy
DPSGD	Differentially Private Stochastic Gradient Descent
FC-NN	Fully-Connected Neural Network
FGSM	Fast Gradient Sign Method
FL	Federated Learning
GAN	Generative Adversarial Network
GDPR	European General Data Protection Regulation
HE	Homomorphic Encryption
IMDb	Internet Movie Database
IT	Information Technology
LDP	Local Differential Privacy
ML	Machine Learning
NN	Neural Network
OM	Occasionally Malicious
PGD	Projected Gradient Descent
PIPEDA	Canadian Personal Information Protection and Electronic Documents Act
RQ	Research Question
SA	Secure Aggregation
SGD	Stochastic Gradient Descent
SMPC	Secure Multiparty Computation
SNR	Signal-to-Noise Ratio
TEE	Trusted Execution Environment
ZK	Zero Knowledge

Chapter 1

Introduction

Secure and private Machine Learning (ML) represents a subfield of the broad field of general ML. It addresses the questions of maliciously attacking ML models or their behavior and disclosing information about the models' private training data, respectively. Given that ML models are widely applied in numerous sensitive tasks and within a broad range of domains, unsurprisingly, both the security and the privacy of these models have become lucrative targets for attackers: Targeting the models and their behavior can allow such attackers to subvert the learning system in order to gain profit of any kind, or to simply cause severe damage to the services relying on the models. Disclosing privacy of the models' training data, in contrast, can result in severe implications for the individuals to whom this data belongs to. As a consequence, studying and assuring both the security and privacy of ML models is an important concern when deploying ML models.

This work, first, provides insights into the current state of secure and private ML in practice. Second, it extends existing research on maliciously attacking the privacy of ML models' training data. Therefore, it investigates a novel attack vector that allows for high-fidelity data reconstruction from ML model gradients. Third, this work studies the trust assumptions that underlie theoretical privacy guarantees and explores the implications for ML privacy when these trust assumptions are not held in practice. Forth, it analyzes what impact the implementation of such privacy guarantees has on the models' robustness against adversarial examples. Finally, this work aims at advocating the importance of a secure and privacy-preserving design of ML methods—in particular when these are applied in real-world scenarios.

1.1 Motivation

ML is a rapidly growing field. Since 2012, in universities, the average rate of students being enrolled in ML-related courses has more than tripled [18, p. 49]. In parallel, several countries have put strategies into place that aim at training a given percentage of their population in the topic [18]. Unsurprisingly, this growth goes hand in hand with the increased application of ML models in a broad variety

of domains. In particular, ML becomes more widely applied to sensitive data in critical use-cases such as health care [84, 151], smart metering [71, 192], or the internet of things [103, 134]. Therefore, it suggests itself that the security of the models and the privacy of their sensitive training data need to be protected.

Since the early 2000s, several governments have put regulations into place to provide legal frameworks around the privacy and security of ML applications. Prominent examples of regulations that aim at protecting the privacy of individuals whose data is being collected include the Canadian Personal Information Protection and Electronic Documents Act (PIPEDA) [39], the European General Data Protection Regulation (GDPR) [188], and California's Consumer Privacy Act (CCPA) [173]. In recent years, these regulations have been complemented by, for example, the European AI regulation [83], or the Government of Canada's Directive on Automated Decision-making [164] that also specify, among others, the requirement to ensure that ML systems operate accurately, *i.e.*, they should be safe and secure.

There exist several possible ways to characterize what it means for an ML model to be secure and private. One possible way of defining the secure operation of ML models is by their robustness against adversarial examples, *i.e.*, manipulations of the data that the ML models are supposed to predict on [179]. Take as an illustrative example an ML model that is trained to recognize road signs. This model is supposed to still recognize a stop sign correctly even if an adversary has performed some small targeted manipulations on it. When it comes to privacy, we require the ML model not to leak too much information about its training data. Thereby, we make sure that an attacker cannot confidently rely on the ML model to disclose potentially sensitive properties of this data. For instance, in the case of an ML model that is used to predict treatment for cancer patients, preserving privacy ensures that an attacker with access to this model is not able to confidently reconstruct (parts of) the training data, or to confidently determine which concrete patients' data the model was trained on.

Correctly implementing the regulations and ensuring that ML models provide protection against the described risks is a challenging task. This is due to an incomplete understanding of the risks and worst-case scenarios, to the closely connected uncertainties about how protective theoretical guarantees are in practice, and to the lacking guidelines on correctly implementing security and privacy in ML models. As a consequence, it is not only important to study the current state of affairs in secure and private ML. Additionally, we have to deepen our understanding of how ML models can be attacked in order to provide adequate protection. Further, we have to investigate how much trust in the ML services and their providers is required for this protection to be adequate. We also need to provide realistic estimates of worst-case vulnerabilities of the ML models to specify our requirements correctly. And finally, we need to understand how we can implement the two goals of security and privacy at the same time.

1.2 Problem Statement

In this dissertation, we seek to provide answers to the following research questions that are relevant when jointly studying secure and private ML.

1. **What is the current state of secure and private ML in practice?**
   Given that the field of secure and private ML is a rather young one, it is of high importance to gain an overview on its state in practice. This does not only serve as a survey of the current condition in the field, but instead it helps to uncover shortcomings, open questions, and hidden problems. It also informs the formulation of relevant future research questions.

2. **Where and why can privacy in ML applications be attacked?**
   There exist some known attacks that target the privacy of ML models' potentially sensitive training data. However, to date, our understanding of where and why privacy leakage in ML occurs is far from being exhaustive. Therefore, it is crucial to further explore the attack vector against privacy in ML and to properly characterize the trust assumptions that underlie our theoretical privacy guarantees. Only then will we be able to meet the theoretical privacy guarantees in practice.

3. **Do ML security and privacy go hand in hand?**
   At last, given that ML models are increasingly applied in highly sensitive domains, we require these models to provide both security and privacy for their training data. This raises the question if the goal of security and the one of privacy in ML are well aligned, and how potential trade-offs between these goals can be characterized. This characterization can form the basis for choosing optimal trade-offs. Finally, it can also contribute to the development of methods that jointly, and by design, optimize for the implementation of both goals.

We believe that providing answers to these research questions is essential for the study of secure and private ML, to strengthen the field, and to shape its future development. Even though, in the scope of this work, it is not possible to address every aspect of security and privacy in ML, we show that our answers to the above-mentioned questions represent a valuable and indispensable contribution to advance research in the field.

1.3 Organization

The remainder of this dissertation is organized as follows. In Chapter 2, we briefly introduce preliminary knowledge on ML and Federated Learning (FL), then we provide the background on secure and private ML that is required for the scope of this work. Afterwards, in Chapter 3, we present a survey conducted to capture the

current state of affairs in secure and private ML in practice and the awareness of ML practitioners about the field. The findings of this survey motivate the necessity to study ML privacy more in depth, which we do at the example of FL in Chapter 4 and Chapter 5. Chapter 4, therefore, first presents our observation on inherent data leakage from Neural Networks (NNs). Based on this observation, we propose a novel highly efficient data extraction attack. Our attack relies on adversarially manipulating the model weights and architecture and then extracting the data from the model gradients. After introducing the attack, we show that it represents a novel attack vector in FL protocols by enabling the server to break the privacy of the clients. Finally, we evaluate our attack's effectiveness in this scenario. In Chapter 5, by building up on our novel attack vector, we study the trust assumptions of FL with a focus on the trust required in the server. Therefore, we do not only examine standard vanilla FL but also several of its extensions. In Chapter 6, we then study how implementing theoretical privacy guarantees impacts the model's security. Therefore, we investigate private models' robustness under the presence of adversarial manipulations against the data that the model is supposed to predict on. To conclude this dissertation, in Chapter 7, we provide a brief summary of the work, highlight our contributions in the area, and present future research directions of secure and private ML.

Chapter 2

Background

This chapter introduces concepts and background knowledge required within the course of this work. It starts by providing an overview of centralized Machine Learning (ML) and Federated Learning (FL). Afterwards, it presents an introduction to the security and privacy of ML. Therefore, it first describes the general threat space against ML models and the systems within which they are deployed. Then, it briefly outlines attacks that can be conducted to target ML security and privacy. Moreover, it gives a brief overview of possible defenses. Finally, a more thorough introduction to the topic of Differential Privacy (DP) [57] is provided due to its relevance for this work. The chapter concludes by presenting the datasets that are used for experimental evaluation throughout this work.

Note: The chapter is not intended to provide a complete survey on the wide field of security and privacy in ML. Instead, it serves as a general overview and reference material for the scope of this work, containing pointers to further literature for interested readers.

2.1 Introduction to Machine Learning

ML is applied for automated discovery of knowledge [112] or regularities, *i.e.*, pattern, [25] in large datasets. Thereby, it enables for automated analyses on those datasets [129]. The following section aims at providing a brief overview on ML, introducing the terms and concepts required for the scope of this work. For a more thorough introduction into the field, see, for example, [25, 129].

2.1.1 Terms and Notation

The following definitions are adapted from [25].

Training. In ML, the process of learning some properties of a dataset $\mathcal{D}_{train}$ is referred to as *training*. The result of the training is a parameterized function $f_{\mathcal{W}}$ whose internal *parameters* $\mathcal{W}$ are adapted to fit the given training data. The training process is therefore also called *fitting*.

Training Data. The dataset $\mathcal{D}_{train}$ used for training is called *training data*. It represents a subset of data distribution $\mathcal{D}$, *i.e.*, $\mathcal{D}_{train} \subset \mathcal{D}$.

Features. The variables characterizing each training data point $x_i \in \mathbb{R}^m$ are called *features*. Each data point has $m \in \mathbb{N}$ such features $\{x_{i,1}, \ldots, x_{i,m}\}$.

Model. The learned function $f_{\mathcal{W}}$ can be referred to as the ML *model*.

Validation Data. The *validation dataset* $\mathcal{D}_{val} \subset \mathcal{D}$ is a separate dataset from the training dataset which is used to optimize the training process via hyperparameters, *i.e.*, internal parameters that control the learning of $f_{\mathcal{W}}$.

Test Data. The *test dataset* $\mathcal{D}_{test} \subset \mathcal{D}$ is another separate dataset consisting of data points not used for the training of $f_{\mathcal{W}}$. $\mathcal{D}_{test}$ serves to evaluate the final performance of the model $f_{\mathcal{W}}$ after training.

Generalization. The trained model $f_{\mathcal{W}}$ should not only fit the training data well, but it should perform similarly well on the test data, which is known as *generalization*.

Overfitting. The opposite of good generalization is called *overfitting*. It describes the effect when $f_{\mathcal{W}}$ performs well on the training data but exhibits a poor performance on the test data.

2.1.2 Supervised Machine Learning and Neural Networks

There exist three broad classes of ML algorithms, namely *supervised*, *unsupervised* and *reinforcement learning* [25]. This work focuses primarily on supervised ML where the training data consists of data points $X = \{x_i\}_{i=1}^n, = x_i \in \mathbb{R}^m$ and their corresponding target labels $Y = \{y_i\}_{i=1}^n$, such that $\mathcal{D}_{train} = \{(X, Y)\}$. The respective target labels can belong to a finite number of discrete categories (*classification*) or to one or more continuous variables (*regression*).

This work focuses on ML models, more precisely Neural Networks (NNs) for classification, hence $y_i \in \mathbb{N}$. We denote an NN classifier by a function $f_{\mathcal{W}} : \mathbb{R}^m \rightarrow \{1, \cdots, k\}$ where $k \in \mathbb{N}$ is the number of different classes encountered in the target labels. The parameters $\mathcal{W}$ in an NN are often referred to as *weights* and they are updated during training.

Training. Training NNs is done via greedily traversing the weight space towards the direction that minimizes an error metric on the training examples, usually called the *loss* $\mathcal{L}$. For a given point in the weight space, and for a given collection of training examples, this direction is given by the model's *weight gradient* G—the vector that points to the steepest loss slope.

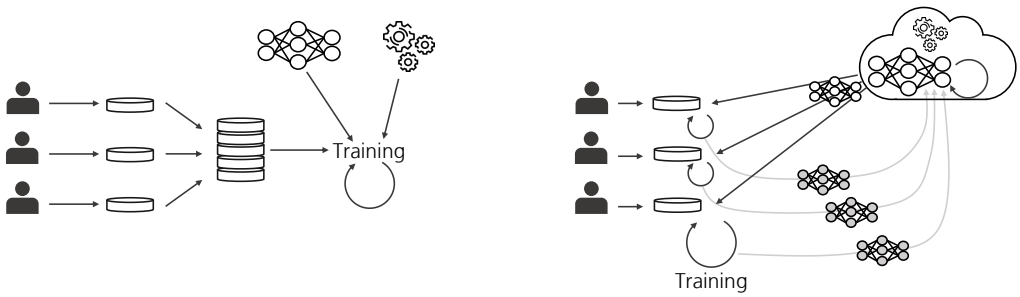
One popular training algorithm for traversing the weight space is the mini-batch Stochastic Gradient Descent (SGD). In mini-batch SGD, instead of processing the

whole potentially large dataset $\{(X, Y)\}$ at once, smaller data mini-batches of a fixed size B are sampled from the dataset. Then, training is conducted using the following steps: (1) sample a mini-batch of size B from the training data $\{(X, Y)_b\}_{b=1}^{B}$, (2) take a forward pass through the model to obtain its predictions on the mini-batch, (3) compute the difference between predictions and ground-truth labels, which yields the *loss* value $\mathcal{L}$, (4) compute the gradient of $\mathcal{L}$ w.r.t. the weights to obtain the *weight gradient* $G = \nabla_{\mathcal{W}}\mathcal{L}((X, Y)_b)$. Then update the model weights according to this weight gradient and a specified learning rate η as $\mathcal{W} \leftarrow \mathcal{W} - \eta G$. Executing mini-batch SGD over all B mini-batches, *i.e.*, the whole dataset, is often denoted as an *epoch*, and can be iteratively repeated for T epochs.

Weight Initialization. Before training can begin, model weights $\mathcal{W}$ have to be initialized. Typically, a randomized initialization is used, *e.g.*, a Gaussian distribution with zero mean, *i.e.*, $\mu = 0.0$ [74], or a distribution specifically tailored for this purpose, *e.g.* the *Xavier* [75] or *He* [89] initialization. Initialization methods have a dramatic effect on learning success [54]. Sub-optimal initialization can lead to vanishing or exploding gradients, or other poor convergence properties. Manipulated model weight initialization controlled by an attacker can cause longer training and reduce the final model utility [81].

Neural Network Architecture and Layer Types. An NN classifier $f_{\mathcal{W}}$ is usually implemented as a sequence of l layers l_i with $i \in \{1, \dots l\}$, each of which has a weight matrix W_{l_i} and a non-linear *activation function* F. A popular activation function is ReLU, given by $\text{ReLU}(a) := \max(0, a)$. Each layer processes its input $\mathcal{I}_{l_i} \in \mathbb{R}^m$ by calculating the weighted sum $W_{l_i} \cdot \mathcal{I}_{l_i}$ of the input, where $W_{l_i} \cdot \mathcal{I}_{l_i}$ denotes matrix multiplication[1]. The activation function is then applied as $F(W_{l_i} \cdot \mathcal{I}_{l_i})$. A fully-connected layer l_i consists of o_{l_i} *neurons*. The layer receives input with o_{l_i} features, multiplies them with a two dimensional weight W_{l_i} matrix of shape $(o_{l_i}, o_{l_{i+1}})$, and applies the activation function element-wise. A convolutional layer consists of multiple two dimensional weight matrices, called *filters*. The input to a convolutional layer is a set of spatially-adjacent features that are multiplied with the filters and put through an activation function to produce the layer's output, called a *feature map*. We refer to NN classifiers that solely consist of fully-connected layers as Fully-Connected Neural Networks (FC-NNs), and NN classifiers that also contain convolutional layers as Convolutional Neural Networks (CNNs).

At the output of a NN classifier both for FC-NNs and CNNs, there is usually a fully connected output layer with a *softmax* activation function which produces a probability distribution over the k possible classes [25].



(a) Centralized Learning. (b) Federated Learning.

Figure 2.1.: **Centralized vs. Federated ML.** In centralized ML, the model's training data is first collected at a central location where the training is conducted. In FL, the data remains distributed locally over different clients. A server sends out a shared model to the clients during the iterative training procedure. The clients calculate model gradients locally, and then send their local gradients to the server. The server aggregates all gradients and applies them to the shared model before sending it out to the clients again.

2.2 Introduction to Federated Learning

Federated Learning (FL) [120] is a communication protocol for training a shared ML model $f_{\mathcal{W}}$ on decentralized data $\{(X_i, Y_i)\}_{i=1}^{\mathsf{N}}$ owned by N different clients $\{u_i\}_{i=1}^{\mathsf{N}}$. Informally speaking, FL implements a decentralized version of the mini-batch SGD algorithm where the data mini-batches are distributed over the different clients. It, thereby, reduces the communication and central storing costs from centralized learning since the training data does need to be sent to the server. See Figure 2.1 for a visualization of centralized vs. federated ML.

For jointly training $f_{\mathcal{W}}$ over the distributed data, in FL, a server C coordinates the training as follows: Let $t \in \{1, \dots T\}$ be the current iteration of the FL protocol. An iteration in FL conceptually corresponds to an epoch in centralized ML. At iteration $t = 0$, $f_{\mathcal{W}}$ is initialized (at random) by the server C. Usually, the server also specifies the architecture and the learning objective of $f_{\mathcal{W}}$. Let $f_{\mathcal{W}}^{[t]}$ be the model with its weights $\mathcal{W}^{[t]}$ at iteration t. At every iteration t, M out of the N ($\mathsf{M} \ll \mathsf{N}$) clients are selected to contribute to the learning. Then, each of the selected M clients u_i obtains $f_{\mathcal{W}}^{[t]}$ from C and calculates the gradients $G_i^{[t]}$ for $f_{\mathcal{W}}^{[t]}$ based on one mini-batch b sampled from their local dataset $(X_i, Y_i)_b$. In other words, the client computes the gradient $G_i^{[t]} = \nabla_{\mathcal{W}} \mathcal{L}((X_i, Y_i)_b)$. Each u_i uploads their gradients to C, who then aggregates all of these gradients to update the shared model's parameters as

[1] We omit bias terms for simplicity; this does not affect correctness of the analysis.

follows:

$$G^{[t]} = \frac{1}{M} \sum_{i=1}^{M} G_i^{[t]}, \quad \mathcal{W}^{[t+1]} = \mathcal{W}^{[t]} - \eta G^{[t]}. \tag{2.1}$$

Note that sharing model gradients is not the only way to perform FL. Other forms of collaboration exist, for example, through sharing model outputs, *e.g.* [48].

2.3 Security and Privacy in Machine Learning

In both centralized and federated ML, recent work highlights that the training process and the final trained ML models are vulnerable to different kinds of attacks. These attacks can target the general *security* of the model and its predictions, *e.g.* [179, 180], or the *privacy* of the model's potentially sensitive training data, *e.g.* [64, 168]. This section aims at providing an overview of the different aspects of ML security and privacy required for the scope of this work. It does not intend to provide a complete survey on ML security and privacy. For this purpose, we refer the reader to existing and more comprehensive work, *e.g.* [14, 142, 158]. Several concepts that represent the main focus of this work are introduced more in detail in their respective sections.

2.3.1 Characterizing the General Threat Space

Analyzing the threat surface of ML and characterizing attackers and their attacks is crucial to reason about ML security and privacy and to propose adequate potential defenses. Therefore, the following section provides an overview of several aspects of the threat surface.

Security Goals. The security goals formulated for traditional Information Technology (IT) security can be adapted to ML as follows [14, 137, 141]:

- *Integrity:* Within an attack to model integrity, an attacker aims at having harmful data points mistaken as benign ones [14]. This induces a model behavior chosen by the attacker [142]. A popular example of these kind of attack is for an attacker to design spam emails such that they bypass a given ML classifier used as a spam filter and are classified as benign emails [14].
- *Availability:* When targeting model availability, an attacker causes the ML system to deny benign data points [14]. With such an attack, the attacker attempts to reduce prediction quality, model performance, or access to the model. If the output of the ML model is involved in the functioning of the system, this can be considered as *denial of service attack* [142]. In the example of the spam filter, an attack against availability aims at having benign emails incorrectly classified as spam—potentially resulting in them not being accessible [14].

- *Confidentiality:* This type of attack can either target confidentiality of the trained ML model or its training data. The former one enables an attacker to obtain sensitive and confidential information, such as model properties, structure, and parameters. Thereby, the attacker may be able to steal the intellectual property represented in the model [142, 180]. The latter one can result in privacy violations if the training data is sensitive.
- *Privacy:* When attacking privacy in ML, an attacker is mainly concerned with disclosing information about the potentially sensitive training data [142].

Integrity and availability are closely related security goals, similar as privacy and confidentiality. Attacks on integrity and availability both operate with respect to model outputs and often rely on introducing unwanted model behavior [142]. Attacks on confidentiality and privacy, on the other hand, operate with respect to the model and the underlying potentially sensitive training data. In particular, a breach of confidentiality can have an impact on data privacy.

Note. A distinction between attacks against confidentiality and against privacy is not necessary if considering the disclosure of information about training data as another aspect of confidentiality-violation. However, semantically the distinction between the two is vital since it enables a more fine-grained specification of the trust model. Whereas attacks against confidentiality threaten the model owner and their intellectual property (an external attacker might steal the model [180]), attacks against privacy threaten the data owners, or users and their privacy [142].

Attacker Knowledge. When attacking ML models, an attacker can have full, partial (to any possible extend), or no knowledge about the following aspects [14, 23]:

1. Training data
2. Training algorithm
3. Trained model and its parameters

An attacker who knows everything about the targeted system is called to have *perfect knowledge* [23]. This setting permits to run the worst-case evaluation of the target model's security [128]. A more realistic setting considers an attacker with *limited knowledge* [23]. Therein, the attacker can be, for example, assumed to know the feature representation of the learning model, but not the concrete training data, or to know the training data, but not the learning algorithm [128].

Attacker Capabilities. The attacker's capabilities can be defined based on the *influence* that the attacker has on the input data and the training procedure [128]. If the attacker can influence the training data or the model training, the attack influence is called *causative*, if they can only exploit existing weaknesses after training, *e.g.* by manipulating the test data [179], it is called to be *exploitative* [14].

Model Access. Tightly connected to the attacker's capabilities is the type of access they have to the ML model under attack. *Black-box* attacks assume that an attacker has no access to the model internals or the training data, but can only interact with the model over a given interface, *e.g.* an Application Programming Interface (API). Through these interfaces, the attacker might use the model as an *oracle* that provides them with outputs for the carefully crafted inputs they provide. In the setting of *white-box* attacks, the attacker disposes of knowledge about the model (algorithm and/or parameters) and/or the training data. The attacker can use the information to identify vulnerabilities in the model and exploit them for attacks [142]. In the real-world, black-box attacks represent a more realistic scenario [142].

Attack Strategies. Barreno et al. proposes to group attack strategies depending on their capabilities along the following three axes [14]:

1. *Influence:* Does the attack target the training time (*causative*) or the testing time (*exploratory*)?
2. *Security violation:* Which security goal(s) are targeted (*integrity, availability, confidentiality, or privacy*)?
3. *Specificity:* Is the attack directed against a particular instance (*targeted*) or of a broader class of instances (*indiscriminate*)?

These axes are independent from each other [14], therefore, there exist $2 \cdot 4 \cdot 2 = 16$ different classes among which attacks and defenses can be grouped. In contrast to viewing the security through the three axes and classification of attacks, one could also look at it with respect to the entire pipeline of ML applications and identify adversarial goals and means at each phase [142].

2.3.2 Machine Learning Security

Based on the definition of attack strategies in the previous section, it is possible to group concrete attacks against ML models. The following provides an overview intended to serve as a high-level orientation. Concrete attacks targeting the privacy of the training data are presented in the following Section 2.3.3.

Training Time Attacks. The most prominent attack against ML models at training time is the so-called *poisoning attack* [24]. In a poisoning attack, an attacker deliberately influences and alters the training data to manipulate predictions of the resulting ML model [142]. This can either be used to reduce general prediction accuracy or to introduce more specific vulnerabilities [112].

Test Time Attacks. At test time, the most prominent type of attacks are *evasion attacks*, during which an attacker aims at causing incorrect model predictions [23]. Therefore, these attacks rely on so-called *adversarial examples* [179]. An adversarial example is a data point that results from perturbing a test data point with a carefully crafted amount of statistical noise before presenting it to an ML model for

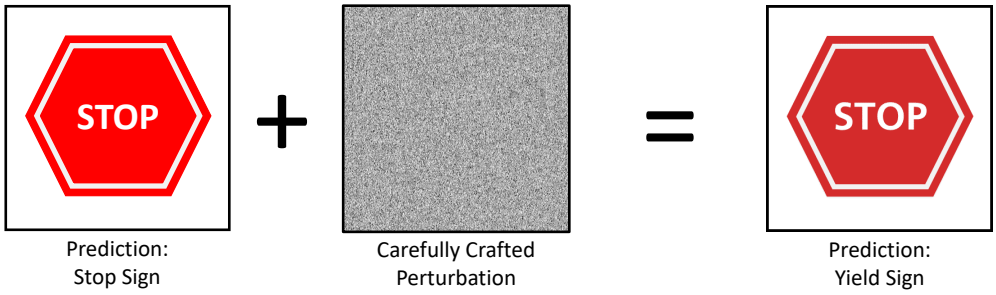


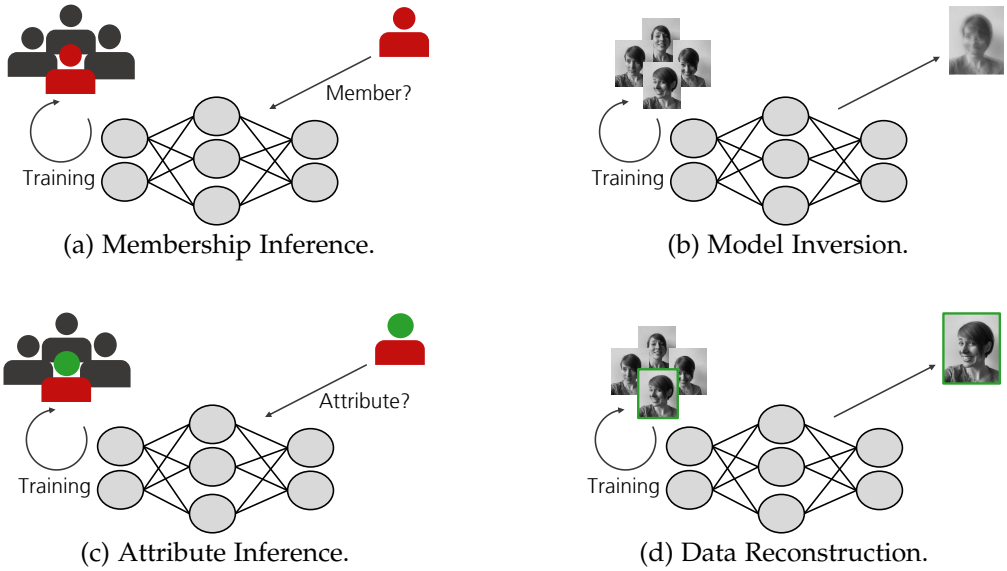
Figure 2.2.: **Adversarial Example.** An adversarial example results from adding a carefully crafted perturbation to a given data point. The resulting difference is nearly invisible, yet it causes the ML model to misclassify the data point.

prediction. Even though the perturbation is hardly noticeable by a human-observer, it causes the ML model to output an incorrect class label. See Figure 2.2 for a visualization of the concept. Adversarial examples can, for example, be used by an attacker to obtain unauthorized access to a system or to slip data through some filters, such as passing spam email through a spam filter [14]. A particular type of evasion attacks is the so-called *impersonation* attack. Impersonation attacks are targeted evasion attacks in which an attacker aims at imitating data points of a specific victim. Thereby, the attacker might, among others, gain the victim's authority in an access control system [112].

Another type of attack against ML models at test time is *model extraction*, also referred to as *model stealing* [180]. An attacker with black-box access to a target model (*e.g.* through an API) can perform model extraction by querying the model with some data points and obtaining labels for this data. With the data and and the received labels, the attacker can then locally train an ML model which reproduces the target model's functionality [180]. The advantage of model extraction over training a different ML model with the same functionality from scratch lies in reduced costs, since the attacker, for example, can skip the expensive manual labelling process of their data [28]. Additionally, model extraction can serve as a basis for further attacks against the security and privacy of the target ML model [180].

2.3.3 Machine Learning Privacy

The topic of ML privacy is mainly concerned with privacy of the potentially sensitive training data of ML models. There exist multiple attacks that aim at disclosing different types and amounts of information about this data. In the following, when presenting the most common types of privacy attacks, we will not distinguish between training and test time attacks as done for the security attacks. This is because, to date, most privacy attacks in centralized ML target the trained



(a) Membership Inference. (b) Model Inversion.

(c) Attribute Inference. (d) Data Reconstruction.

Figure 2.3.: **Privacy Attacks against ML Models.** Overview on different attacks targeting the privacy of an ML model's sensitive training data.

ML models at test time to disclose privacy of the potentially sensitive training data. Only a few instantiations of the attack types described in the following extract private information from gradients during a continuous training process [161], manipulate the training itself [117], or poison the training data [116] for improved privacy disclosure at test time. We visualize the concepts of the privacy attacks introduced in the following in Figure 2.3.

Membership Inference. Membership inference attacks were first introduced in the area of ML in [168]. Given a trained ML model and a data point, they aim at disclosing whether the data point was part of the model's training data. This information can be privacy-disclosing when membership itself represents a private information. A practical example of such a scenario could be an ML classifier that predicts treatments for cancer patients. Such a classifier necessarily needs to be trained on data that belongs to individuals who have cancer. The information that an individual is a member of the model's training data, therefore, equal to the information that this individual has cancer.

Model Inversion. Another form of privacy leakage can occur due to model inversion attacks introduced by [64]. These attacks reconstruct average per-class representations of the model's training data. This becomes a privacy leakage when each class corresponds to exactly one individual, such as in face classifier [64], or in speaker recognition systems. In such cases, model inversion discloses the identity of the individuals that the model was trained on.

Attribute Inference. The term attribute inference is sometimes used to refer to an attack aiming at the disclosure of some properties of the training dataset, such as the distribution of a sensitive feature among the training data points [8]. In this work, however, we follow the naming introduced in [64] and use the term *attribute inference* to describe an attack where the attacker uses knowledge on some public attributes of a training data point and access to the trained ML model with the aim of disclosing some sensitive and secret attribute of that data point. In the example of the cancer treatment classifier mentioned above, an attacker could use an individual's age, gender, weight, and height, to disclose a sensitive biometric attribute, such as this individual's blood group.

Data Reconstruction. A particularly severe form of attacks against privacy of an ML model's training data is data reconstruction [70, 148, 161, 193, 200, 205, 207]. The attack aims at reconstructing individual training data points from a trained ML model or from model gradients by performing some reconstruction procedure. A specific form data reconstruction is *data extraction* [30, 63], where the individual training data points can be extracted directly and without any error from the ML model or its gradients.

2.3.4 Defending Machine Learning

The following section provides a high-level overview on possible defenses against security and privacy risks in ML. It does not aim at providing a comprehensive survey nor at presenting a thorough analysis of and comparison between different defenses for different attacks. Instead, it briefly introduces the concepts relevant for the scope of this work.

Generally speaking, defense strategies in ML aim at preventing successful attacks against the models or the systems that surround them. Most defenses are specific to a particular class or type of attack. In the following, we provide an overview on two broad types of defenses, namely those defenses that aim at protecting the system around an ML model, and those that mainly aim at protecting the model itself. Of course, it is also possible to combine multiple of the defenses described in the following.

2.3.4.1 Defending Machine Learning Systems

Data. Since ML operates on data, the first possibility of protecting the security of an ML system is to ensure integrity of its training data. This can be done, for example, by implementing *data provenance* [36, 47]. Data provenance allows to track where a data point in a given dataset stems from and how it was processed [36]. This is of high importance since the characteristics of an ML model's training dataset have a substantial impact on the model's behavior [69]. Yet, modern ML systems do not necessarily operate on carefully and centrally curated training datasets anymore

but rather on large amounts of data constantly generated by many users. With the lack of a centralized instance controlling data integrity, the quality of data can vary widely, and therefore, tracking data provenance and additional contextual information can help judging whether a data point is trustworthy [47]. This can, for example, be beneficial for protecting against data poisoning attacks. Additionally, data provenance can also contribute in the mitigation of the risks resulting from biases, hidden underlying assumptions in the data, or mismatches between the training and test data distribution [69].

When it comes to privacy protection, a popular protection approach within the ML system is *data sanitization* [136]. Data sanitization describes the process of transforming an original dataset into a sanitized one [136]. Within the transformation, sensitive information inside the dataset is removed or hidden while preserving the statistical properties of the dataset. This allows for balancing out the need privacy protection and the ability of performing meaningful analyses [136]. Another way for protecting privacy is to store only the data required for a particular analyses, and—in case that data from different analyses needs to be combined—rely on techniques such as *privacy preserving record linkage* [85, 186]. Privacy preserving record linkage allows to combine data concerning the same individual held by different parties without these parties having to reveal additional information to each other, in particular informaion about other individuals whose data they hold [85].

System. When it comes to defending the system itself, both in terms of security and privacy, methods from traditional information security and *system security* [172] can be applied. For example, *access control* [163] to the data and ML models can protect against malicious manipulations that target system's integrity. At the same time, access control can prevent disclosing sensitive information or sensitive training data to unauthorized parties. Closely related is the concept of *separation of privileges*, or *least privileges* [3] which specifies that each party can solely access and alter parts of the system required for the fulfillment of their specific tasks. These concepts around privileges are not necessarily restricted to human parties involved in the ML system. Instead, it is also possible to extend them to system components, *e.g.* by denying an ML model access to its training data, once its training is completed [137].

However, not all insights from traditionan information security are applicable directly to ML [137]. For example, in traditional information security, combining different computer systems with various architectures introduces complexity and additional risks. In ML systems, in contrast, Papernot *et al.* [137] point out that combining different models, for example through *ensemble learning* [55], *i.e.*, combining the predictions of multiple (heterogeneous) ML models [160], can be beneficial for security. This is because even if a subset of these models is successfully attacked, the models' combined prediction can still be correct and of integrity.

Test Time. At test time, *i.e.*, after training when the ML model is exposed to perform predictions on new and unseen test data, there exist additional defenses. One possible defense relies on *observing the model input (and output)* [15]. This allows to identify anomalies or potential deviations in the input or output distributions, and thereby, to detect attacks against the ML model. To limit the success of attacks such as model stealing, another possible defense relies on *introducing delay* [167] in the model responses to queries. Thereby, the knowledge on the model and its private training data that an attacker can extract in a given amount of time can be limited. A technique that increase ML models' security against adversarial examples is smoothing [50]. Therefore, the input data at test time is perturbed with statistical noise and the ML model's prediction on the perturbed data is returned [50].

2.3.4.2 Defending Machine Learning Models

Another protection method which secures the ML models directly against the impact of adversarial examples is the so-called *adversarial (re-)training* [78]. Therefore, the ML model's training data is extended by adversarial examples and the model is trained on the resulting combined dataset.

When it comes to protecting ML models against stealing attacks, one popular approach consists in *watermarking* [184]. Within this form of intellectual property protection, a model owner introduces some unusual characteristics to the model parameters [184] or the model's prediction behavior [203] which allow the owner to detect potentially stolen copies of their model. For a systematic review on ML model watermarking, we refer the reader to [28].

When it comes to protecting confidentiality and privacy of the ML model's potentially sensitive training data, the most popular approaches include *Homomorphic Encryption (HE)* [79], *Secure Multiparty Computation (SMPC)* [111], and *Differential Privacy (DP)* [57]. HE allows to perform mathematical operations on encrypted data such that the decrypted result corresponds to the result that is obtained when performing the same mathematical operation on the unencrypted data. In ML, HE provides building blocks to perform training [6] and predictions [154] on encrypted data while preserving data confidentiality [79]. SMPC is a cryptographic primitive that allows distributed parties to jointly compute an arbitrary functionality without revealing their private inputs or outputs to other parties [206]. In ML, SMPC has two main applications: first, it enables training on data which is distributed across several data sources, usually by providing a data aggregation scheme for encrypted data. Second, it allows to implement confidentiality for both model and data when the trained ML model and data at test time are owned by different parties. Therefore, it provides primitives to perform predictions such that the party holding the model does not learn anything about the other party's data, and the party owning the data learns as little as possible about the models [206]. DP is a framework that allows to learn some statistical properties of a dataset without disclosing sensitive

information about the individual data points within the dataset [57]. Therefore, statistical noise is added to the algorithm processing the dataset. In ML, the noise is usually introduced during the model training to preserve privacy of the training data [2, 138]. The following section introduces the concept of DP more formally due to its relevance for the remainder of this work.

2.3.5 Differential Privacy

The framework of DP [57] formalizes the intuition that no single data point should have a significant influence on the results of an analysis conducted on a whole dataset. This enables learning properties of the dataset and the data population represented within this dataset while preserving privacy of individual data points. To achieve this goal, an analysis under DP should yield roughly the same results whether or not a particular data point is included in the dataset that the analysis is performed on. For a more thorough introduction to DP, we refer the interested reader to [58].

2.3.5.1 General Differential Privacy

Formally, the concept of (ε, δ)-DP can be expressed by the following definition.

Definition 2.1 ((ε, δ)-Differential Privacy)**.** Let $A \colon \mathcal{D}^* \to \mathcal{R}$ a randomized algorithm. A satisfies (ε, δ)-DP with $\varepsilon \in \mathbb{R}_+$ and $\delta \in [0, 1]$ if for all neighboring datasets $D \sim D'$, *i.e.*, datasets that differ in only one data point, and for all possible subsets $R \subseteq \mathcal{R}$ of the result space

$$\mathbb{P}\left[A(D) \in R\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[M(D') \in R\right] + \delta\,. \tag{2.2}$$

The parameter ε which is often referred to as *privacy budget*, or *privacy level* bounds the maximal difference between the analysis results on the neighboring datasets [57]. Smaller values for ε correspond to higher levels of privacy, whereas larger values imply lower levels of privacy. The second parameter δ represents a relaxation of the bound by allowing the results to vary more than the factor e^{ε}. Hence, the total privacy loss is bounded by ε with a probability of at least $1 - \delta$ [58].

2.3.5.2 Differential Privacy in Machine Learning

While there also exist other forms to integrate DP into ML, *e.g.* [138], in the scope of this work, we will focus on the de-facto standard algorithm, namely Differentially Private Stochastic Gradient Descent (DPSGD) [2]. Similar to general DP, the DPSGD implements the intuition that no single training data point can significantly impact the resulting ML model. Therefore, DPSGD alters the standard SGD-based training process for ML models to introduce DP guarantees into the weight updates. This is done through two subsequent steps. In the first step within DPSGD, the gradients

are computed for each data point or, to make training more efficient, each mini-batch of data points. The gradients are then clipped such that their ℓ_2-norm does not exceed a pre-defined clipping parameter c. More precisely, the clipping of the gradient $G = \nabla_{\mathcal{W}}\mathcal{L}((X,Y)_b)$ for mini-batch b of some data (X,Y) is clipped by replacing G with $\bar{G} = G/\max(1, \frac{||G||_2}{c})$. Thereby, if $||G||_2 \leq c$, the gradient will not be altered, whereas if $||G||_2 > c$, the gradient will get scaled down to norm c. In the second step, Gaussian noise with scale σ is applied to the clipped gradients $\bar{G}$ of each mini-batch before performing the model updates. This results in the gradient update being performed with $\bar{G} + \mathcal{N}(0, \sigma^2 c^2 \mathbb{I})$ [2]. The matrix $\mathbb{I}$ is the identity matrix with dimensions respective to the gradient dimensions. In the following, we are going to suppress this matrix when referring to the noise added under DP.

2.3.5.3 Differential Privacy in Federated Learning

There exist three main ways of integrating DP in the FL protocol.

1. *Centralized Differential Privacy (CDP)* assumes a trusted server that implements the privacy mechanism, for example, through an adaptation of the DPSGD [2]. Therefore, the clients clip their gradients locally and the server performs the addition of noise [152]. To improve utility and leverage privacy amplifications in the CDP setup in FL, techniques, such as random check-ins [13], or the DP-FTRL mechanism [100] where clients are sampled at random were introduced. However, as we will argue in Chapter 5, CDP cannot provide DP guarantees when assuming a malicious server that does not perform the privacy operations, or extracts data before adding noise.
2. *Local Differential Privacy (LDP)* was proposed to resolve the need for trust in the server. In LDP, every client locally adds noise to their gradients according to their privacy requirements [182]. Independent of other clients, the noise is drawn from $\mathcal{N}(0, \sigma^2 c^2)$. However, previous work has shown that this setup leads to poor privacy-utility trade-offs, such that LDP is not popular in practical applications [103].
3. *Distributed Differential Privacy (DDP)* is supposed to combine the advantages of CDP and LDP. In DDP, before aggregation, each client locally adds some (small) amount of noise to their gradients [181]. The noise distribution depends on the number M of other selected clients. It is specified by $\mathcal{N}\left(0, \frac{\sigma^2}{\mathsf{M}-1} c^2\right)$ [181]. While the individual noise levels don't offer sufficient protection, the aggregates provide rigorous privacy guarantees. There exist different forms of performing the aggregation. One of the most popular approaches is an SMPC protocol called Secure Aggregation (SA) [33].

Secure Aggregation. SA was developed to protect the individual $G_i^{[t]}$ from inspection by the server [33]. In SA, instead of sharing the individual gradients with the server, an aggregate over all M clients' gradients is computed, and only the final outcome $G^{[t]}$ of the computation is shared with the server. Extensions of SA, among

others, allow the server to prove the correctness of the aggregate computation [197], increase robustness against malicious updates [37], and improve communication efficiency [17, 82].

Note that the application of SA increases the computational, storage, and communication costs in FL—several costs grow quadratically with the number of clients [32, 33]. This limits the maximum number of clients that can participate in each round of the protocol. Furthermore, SA protocols rely on integer inputs. However, usually the gradients calculated, as well as the noise added to implement DP guarantees, are real-scaled values. Therefore, standard gradient calculation and DP-mechanisms cannot be applied. Instead novel mechanism to perform gradient aggregation and noise addition with integer values were developed *e.g.* [5, 99]. Even though there exist extensions to make such mechanisms more communication efficient [46], their practical applicability remains, as of now, limited.

2.4 Datasets

To perform extensive experimental evaluation of the methods proposed in this work, we rely on several vision and one text-based datasets. These datasets are described in the following.

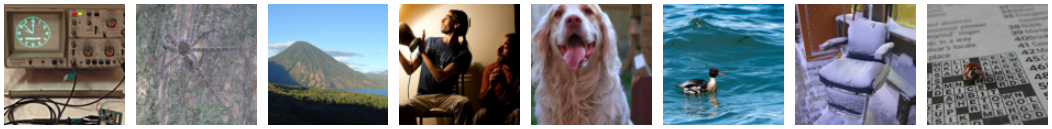
MNIST

The MNIST [110] dataset is a vision dataset consisting of 70,000 gray-scale images and corresponding labels for ten classes. The images are of size 28x28 pixels and depict the hand-written digits zero to nine. The dataset is divided into a training set consisting of 60,000 images and a test set consisting of 10,000 images.

CIFAR

CIFAR [106] datasets consist 32x32 color images that depict different objects or animals. The *CIFAR-10* dataset in total holds 60,000 images, 50,000 for training and 10,000 for testing. The images belong to ten different classes with 6,000 images per class. The larger *CIFAR-100* dataset, as suggested by its name, contains 100 classes with 600 images per class, 500 of which for training and 100 for testing.

ImageNet



ImageNet [53] is a large and complex vision dataset containing color images organized according to the WordNet hierarchy.[2] Within the scope of this work, we use the *ImageNet Large Scale Visual Recognition Challenge*-subset. It contains 1,281,167 training, 50,000 validation, and 100,000 test images. The complexity of this dataset is mainly due to its high dimensionality and the large number of classes.

Internet Movie Database

> *"A wonderful little production. [...]"*, (positive)
> *"An awful film! [...]"*, (negative)

The text-based Internet Movie Database (IMDb) [114] dataset is used for binary sentiment analysis. It contains 50,000 reviews from the IMDb and corresponding labels 'positive' and 'negative'.

[2] https://www.image-net.org/about.php, last accessed on July 16, 2022.

Chapter 3

Surveying Secure and Private Machine Learning in Practice

With Machine Learning (ML) components being increasingly applied within sensitive applications, such as health care [84, 151], smart grids [71, 192], and hiring processes [43], ensuring security and privacy of the systems rapidly gains importance. Yet, Section 2.3.2, and Section 2.3.3 in the previous chapter highlight that there exists a myriad of attacks targeting different aspects of the ML security and privacy. Luckily, there also exist corresponding defenses, as discussed in Section 2.3.4. However, the pure existence of defenses does not result in protected systems since these defenses need to be actually (and correctly) implemented.

The gap between theoretical existence and practical implementation of secure and private ML motivates the necessity of studying the state of affairs in secure and private ML in practice. By studying the security and privacy risk-awareness, the practical prevalence of defense methods, and the difficulties encountered when implementing them, we can inform future development of novel protection methods and tools. Such a study, furthermore, helps advancing the field of secure and private ML by providing an enhanced understanding of the risks and their practical implications, current shortcomings in defenses, and improvements that future research will have to provide.

To address this important topic, and to gain insights into current secure and private ML in practice, we set out conduct a study among ML practitioners, *i.e.*, individuals implementing ML systems. Our aim was to learn about their awareness on risks and defenses, and about their current practices.

In this chapter, we present our study which is based on an online survey conducted in two phases among ML practitioners from different countries. Studying ML practitioners is of particular importance since they are the individuals in charge of putting theoretical advancements in research and regulatory requirements formulated by governmental agencies around security and privacy in ML into practice.

Yet, in prior research, neither these individuals nor the practical state of affairs in secure and private ML have received a lot of attention.

Our study aims at closing this gap and attaining the following three goals:

- Understanding the current state of awareness;
- Identifying factors that influence awareness among ML practitioners;
- Exploring the actual use of existing tools and methods for secure and private ML in practice.

In this chapter, we, therefore, first introduce our study's research questions. We then provide an overview on related work in the area of studying security and privacy practices. Afterwards, we present in detail our research methods and results. We conclude with a short summary, a discussion of our findings, and an outlook on relevant future work in improving security and privacy in ML. The work presented in this chapter has been published in [29].

3.1 Motivation and Research Questions

Recent research highlights that the topics of security and privacy, in general, as well as in ML-related workflows, are no significant driving factors when designing new products [108]. Instead, the design of such new products is functionality-driven [132, 133]. Similarly, most research in the field of ML still focuses on proposing novel algorithms with higher performance and extended functionality. Only in recent years has the field of private and secure ML seen an upsurge with the introduction of several techniques that allow for secure and privacy-preserving analyses [29].

In practice, however, the security and privacy of an ML system do not solely rely on the existence of secure and privacy-preserving ML methods. Instead, they largely depend on the actual implementation of these methods. The entire ML system, from the data pipeline and model architecture, up to the concrete implementation and deployment of the final trained ML model are largely determined by the ML practitioners in charge of implementing the systems. Hence, these practitioners and their *awareness* of existing risks and defenses play a vital role when it comes to actual security and privacy in ML. According to [86], there are several definitions of the term awareness in the context of information security. These include but are not limited to an individual's knowledge about risks and defenses, and the importance given to the topic [86]. More concretely, if the ML practitioners are not aware of risks or defenses or do not consider the topic important, the resulting ML systems might lack security or privacy even though adequate defense methods exist.

To the best of our knowledge, this work, published as [29], is the first to study the individual awareness and practices of ML practitioners concerning ML security and privacy. Thereby, it identifies challenges and shortcomings and helps to inform the design and extension of current tools and existing methods.

Within our study, we formulate two Research Questions (RQs):

1. *RQ1:* How is ML security and privacy awareness built, and which conditions contribute to the degree of knowledge with respect to threats and corresponding defenses?
2. *RQ2:* What is the current state of affairs concerning ML security and privacy among ML practitioners?

For *RQ1*, we set out to study different aspects of awareness and knowledge acquisition among ML practitioners. To answer *RQ2*, we study the prevalence of different attacks and defenses against ML security and privacy. We, furthermore, survey the practitioners' practices and experience with selected standard libraries for secure and private ML. Finally, based on the example of the European General Data Protection Regulation (GDPR), we also investigate the influence of the introduction of juridical regulations on the practitioners' practices. Our research is supposed to close a critical gap on the way towards bringing theoretical research on secure and private ML into practice.

3.2 Background and Related Work

Studying security and privacy practices in ML separately from practices in standard software applications is of high importance. This is, first of all, because ML exhibits a different threat space and therefore has different requirements for defenses than standard software applications [137]. Second, well-established practices from standard software security, such as static code analysis or code coverage, are not directly applicable to ML models. This is due to the fact that the behavior of an ML model does not solely depend on the code used to train or deploy it. Rather, it is largely influenced by the model's training data. As outlined in Chapter 2, the data, thereby, has a large influence on the model's functionality and can even introduce vulnerabilities, for example, through poisoning attacks. Despite the topic's high importance, there exists very limited work on studying the security and privacy practices in ML, in particular with a focus on the individuals in charge of implementing the systems.

This chapter provides some background on studying security and privacy practice among software developers. Additionally, it presents relevant studies in the field—with and without a focus on ML—to put our work into context. For background on the security and privacy attacks in ML and the corresponding defenses that served as a guide to developing the survey questionnaire for our study, we refer the reader to Chapter 2.

Studying Developers. Studying developers, *i.e.*, individuals in charge of designing, implementing, or maintaining program code, is a challenging and, so far, not well established task. Therefore, studies about developers usually have small sample sizes or rely on university computer science students [4]. Current research [21,

91, 162, 178] suggests that it is valid to rely on students within the scope of such studies. Salman *et al*. [162], to name an example, compared students and developers for (non-security-related) tasks. Their findings show that in case students and developers have roughly the same level of experience, the quality of the code they produce is comparable.

Studying Developers' Security and Privacy Practices. Within the research field of studying developers in general, there is a small body of literature related to explicitly studying developers' security and privacy practices.

Acar *et al*. [4] studied Python developers in form of an online study. They aimed at investigating the impact of applying security Application Programming Interfaces (APIs) on the security level of the resulting code. Therefore, the authors had the developers perform specified coding tasks and fill in an online questionnaire. The results of the study showed that years of experience are an influential factor in the functionality and security of the code whereas the self-reported status (professional or student) is not. With each year of experience, the authors measured a 10% increase in the likelihood of the developers' code being functional and a 5% increase in the likelihood of the code being secure. Yet, for both students and developers, the general observed security level of the code was low.

Naiakshina *et al*. performed a coding task study with 20 [132], and a survey with 40 [133] computer science students, respectively. The authors concluded that participants usually consider functionality before security. Additionally, the resulting security level of the code only increases when the security requirements are explicitly stated to the participants. Furthermore, the authors showed that knowledge of security practices does not necessarily lead to their implementation in practice. Finally, the results suggest that the existence of security features within APIs is not sufficient to guarantee the security of the resulting code if these features are activated in an opt-in fashion. Therefore, the authors argue that the security features in such APIs should be the default option.

Additional research in the area of security and privacy practices of developers specifically targets app development.

Balebako *et al*. [12] conducted a survey about privacy practices and awareness among 200 app developers. Their main findings highlight that, in general, developers find privacy policies hard to read. Additionally, the developers express criticism about such policies mainly being created without input from individuals working in development. Moreover, the majority of developers states that they have learned about privacy practices only once being confronted with tasks requiring their implementation. Then, the developers rely on social networks and experts around them for help on the implementation and application of such practices. Most developers also never received formal training on privacy practices. Another finding of the

authors suggests that also in app development, functionality is considered before privacy. Finally, the individual developers' awareness of privacy does not seem to be a good indicator of the quality of their practical implementation of privacy practices.

Such findings have motivated the emergence of an area of research called *developer-centered security*. Research in this field suggests that writing security related code is not something the average software developer deals with on a regular basis, nor is security education a strong focus at many universities teaching computer science [4]. Therefore, when putting usable security into place, it should not be focused purely on end users, but also on developers (*i.e.*, end users of security APIs or libraries) [40, 80, 149, 174, 196].

In this vein, Nadi *et al*. [130] studied Java developers and their use of crypto APIs. They found that through good documentation within the API, the developers' code security could be significantly improved. Similarly, Jain and Lindqvist [94] were able to show that by providing appropriate APIs, developers can be nudged into choosing more privacy-preserving coding choices.

Studying ML Practitioners' Security and Privacy Practices. While the previously mentioned related work focuses on studying security and privacy practices among the general group of developers, our study focuses on ML *practitioners*. A definition of the term ML practitioner and its delimitation to the term of developer used within related work are provided in Section 3.3.3. Since the field of secure and private ML is highly specific, the pool of ML practitioners is even smaller than the pool of general developers. This makes studying their practices a challenging task. To the best of our knowledge, this work, published as [29], is the first and only one studying individual ML practitioners' security and privacy awareness and practices.

Studying ML Security and Privacy within Companies. In the work that is closest to our study, Kumar *et al*. [108] studied ML security and privacy in practice by surveying companies that rely on ML. More precisely, the authors conducted interviews with two employees responsible for creating ML models and the security personnel responsible for securing the company's infrastructure from 28 different companies. Their work, thereby, differs from ours in terms of study focus. While we focus on the individuals in charge of implementing ML systems, and their awareness and perception concerning the importance of the issue, [108] considers the topic from a higher level, namely an institutional one with a particular emphasis on the workflows. Their results highlight that most companies still rather put an emphasis on traditional IT security. In general, the companies do not seem to have the tools or knowledge to protect their ML systems. Furthermore, the study reveals that privacy attacks are considered particularly threatening by companies. While [108] work is mainly concerned with revealing gaps in the security of

technical workflows, our work focuses on reasons and influential factors leading to these gaps among the individuals in charge of the ML systems' implementation.

3.3 Method

We conducted our study of awareness and practices in secure and private ML among ML practitioners from all over the world. Therefore, we relied on an online questionnaire containing questions about their perception and practices of ML security and privacy along with questions on our participants' demographics.

We split our study into two separate phases, a preliminary pilot and the full study. The pilot study was intended as a validation instrument for the survey and to identify potentially missing aspects that could then be added to the final study.

To host our online questionnaire, we used LimeSurvey [76]. Participants were not compensated for their participation. Each participant was informed about the handling of the data collected throughout the participation, and about the fact that participation is voluntary and can be discontinued at any point. Solely the participants consenting to data collection were forwarded to the online questionnaire.

3.3.1 Pilot Study

For our pilot study that was limited to participants in Europe, we sent out a link to our online questionnaire via email to 531 AI-related companies in Europe. In total, 41 ML practitioners from eleven different countries completed our questionnaire. Additionally, the questionnaire was given out to 40 students enrolled in an ML-related course at Free University Berlin. 32 out of the 40 students completed the questionnaire. For detailed demographics on both developers and students who completed the questionnaire, see Section A.2.1.1 in Appendix A.2.

The developer questionnaire consisted of 39 questions and took 16 min. 58 sec. on average to fill out. For the student questionnaire, we left out the last section about the GDPR and changed the questions in the demography group. The student questionnaire consisted of 26 questions and took 7 min. 37 sec. on average.

We relied on the pilot study to obtain an accurate impression of the current state of affairs in secure and private ML in practice. This can be considered an evidence-based-design approach: instead of including solely information from existing research papers, we heavily relied on quantitative elements in our pilot study in form of free-text fields to inform the design of our full study. Relying on free-text fields allowed us to gain qualitative insights into the ML practitioners' perception, experience, and practices around secure and private ML and privacy regulations at the example of GDPR. Such aspects and details about individual perception would not be fully captured through closed questions.

Additionally, using free-text fields can help to mitigate biased answers, such as, when a participant is aware of an attack or defense against ML security or privacy but does not know the name used to refer to it within the questionnaire. Therefore, we asked participants to describe, using their own words, what risks and protective measures around the security and privacy of ML models there were aware of. For the selection of ML libraries, we proceeded in the same way. Through additional free-text fields, we collected further comments from our participants, for example, on the question of whether our answer options were understandable and complete.

To evaluate the free-text answers, two researchers separately identified code books for different question groups. After an agreement on the final code books was reached, both researchers independently applied these to the participants' answers. By using Cohen's Kappa for inter-rater reliability [49], the agreement of the two raters was calculated. This resulted in a Kappa value of 0.93, indicating strong agreement on codes and labels in the data [62]. Finally, each case of disagreement was resolved independently by discussion.

In addition to the free-text fields, our study also contained quantitative elements such as multiple choice questions. These aimed at finding patterns in the data and drawing conclusions on relationships between participants' demographics, security and privacy awareness in ML, and their resulting practices. Both our student and developer questionnaire, as well as the code books, can be found in Section A.2.1.2 in Appendix A.2.

3.3.2 Structure of the Final Questionnaire

Informed by the pilot study, we adapted the questionnaire for our full study. In total, the final questionnaire consisted of six main question groups. Their description is taken from [29]:

1. Demographics: This section captures the demographic background of the participants, *i.e.*, their education, the country they were working in, their daily ML-related tasks, and their present working situation. Participants who indicated that they were currently employed were asked additional questions about their employing company, *e.g.* number of employees, how ML is applied, and the sector in which the company operates.
2. Data and Sensitivity: The practitioners were asked, among other things, what type of data they were dealing with, whether this data is (directly) related to individuals, as well as which domain it stems from.
3. ML security: The questions cover how important the participants judged securing their ML models, how they built their knowledge on ML security and privacy, and who in their working environment is responsible for securing the ML models.

4. Attacks: In this question group, four attacks (inversion [65], impersonation [7, 112], poisoning[24] and evasion attack[22]) on the security and privacy of ML models were presented. For each of the attacks, the participants were asked to indicate whether they were familiar with this attack and whether they had already implemented preventive measures to defend against the respective attack. To avoid participants mistakenly marking an attack as unknown just because they were unfamiliar with the particular keyword, a short explanation of the attack was provided together with its name.
5. ML privacy and security practices: Within this section, first, eight ML privacy and security libraries were presented and the participants were asked whether they were familiar with these libraries and whether they had used them. Furthermore, 14 security and privacy practices identified within the participants' answers in the pilot study were presented, along with an explanation for each of them. Again, the practitioners were asked to specify per method, whether they were familiar with it, and whether they had already implemented it.
6. GDPR: The last section contained questions regarding participants' familiarity with the GDPR and the changes in their ML-related privacy practices caused by its adoption.

In total, the questionnaire contained 25 questions and took participants, on average, 11 min. 18 sec. to fill out. The full study's questionnaire can be found in Section A.2.2.1 in Appendix A.2.

3.3.3 Participants

The full study was conducted between July 2020 and October 2020 with ML practitioners world-wide. Participant recruiting was performed by promoting the questionnaire over official social media channels and websites of Fraunhofer AISEC, Fraunhofer SIT, and Free University Berlin. Additionally, to reach more international participants, the link to the questionnaire along with a brief description of the research project was posted in ML-specific groups on Reddit and LinkedIn. While the questionnaire was online, in total 1471 individuals clicked the link and opened the survey. Of these, 94 completed the full questionnaire. For the full study, all participants who indicated being a student were filtered out from the dataset, in order to report solely data about actual ML practitioners. This resulted in a dataset consisting of 83 fully completed questionnaires.

The majority of participants held high educational degrees. 80 out of 83 (96%) participants had at least a bachelor's degree. This is consistent with findings of a survey among data scientists conducted by Kaggle (91%) [98].

Out of our 83 participants, 73 (88%) were currently employed, 49 (59%) of which at an early stage of their career measured in years of working experience with

ML. Similar trends were also observed in the Kaggle study [98] where 55% of the participants were reported to have less than three years of experience in the field.

Furthermore, most of our participants reported working for larger companies (54, 65% in companies of over 200 employees). The others worked in medium-sized companies (18, 22% in companies with 11-200 employees), or smaller ones (5, 6%, companies with ten or fewer employees).

Our participants specified working mainly in domains related to *customers and users* or *smart environments and connected devices*. Most of them, furthermore, stated working with the data types *images*, *sensor*, *tabular*, or *text data*. Some participants used the possibility to specify 'other' data types and specified working with *industrial and manufacturing data*, *publicly available datasets*, or *education-related data*.

More than half of the participants indicated ML being the main component of products developed within their department (47, 56.7%). In contrast, one-third declared ML being included in the products but not as a key element (28, 33.7%).

The participants' answers show that using standard ML libraries such as TensorFlow [118] and Scikit-learn [145] is part of the daily tasks of 59 (71%) of them. 54 (65%) participants indicated performing data analyses. For roughly half of the participants, their daily tasks include data evaluation (47, 56.6%), data cleansing and preparation (45, 54%), coordinating ML projects and workflows (44, 53%), as well as developing custom ML applications, *e.g.* designing custom Neural Networks (NNs) for given tasks (36, 43.4%). Note that questions on daily tasks were posed in a multiple choice fashion such that participants could choose all applicable answers.

For a complete overview on participants' demography, background, and working environment, see Section A.2.2.2 in Appendix A.2.

Developers vs. Practitioners. While related work uses term of *developers*, this work considers the slightly broader group of ML *practitioners*. The term practitioner, thereby, refers to all individuals that conduct daily tasks around ML. From our participants' answers regarding their daily tasks, we were able to define the subgroup of 'ML developers' ex post. We consider an ML practitioner to be an ML developer if their activities include 'develop custom ML applications (*e.g.* designing custom NNs for a given task)' or 'develop ML tools or libraries from scratch'. We chose this approach of an ex post definition to counteract the fact that some ML practitioners might not consider themselves a developer when being asked in the questionnaire. This is because there exists no clear definition of the term 'developer'.

3.3.4 Data Analysis Methods

We implemented the data export, preprocessing and analysis in Python [185], using the Python-libraries *scipy.stats* [187] and *factor_analyzer* [165].

Correlation Analysis. To identify correlations within the data, we relied on the *Spearman's rank correlation coefficient*. To identify significant relationships between two categorical variables, we employed the χ^2 *test of independence* [119] We performed group comparisons to identify whether particular response variables and the differences between them can be associated with the membership of a practitioner in a specific group. To compare three or more groups, we used the *Kruskal-Wallis H test* [107]. To compare two groups, or when significant differences between multiple groups were encountered, we performed (additional) pairwise comparisons with the *Mann-Whitney U-test* [60].

Corrections. We relied on the *Benjamini and Hochberg correction* [20] to counter the problem arising from performing multiple comparisons on a single data set. The correction was applied within individual hypothesis families. See the two hypothesis families used during the analysis in Section A.2.2.3. In the following section, when presenting the results of our study, the *corrected p–value*, p^* instead of the original uncorrected p–value p is indicated along with the corrected test statistic.

Exploratory Factor Analysis. To investigate how ML practitioners built their awareness around secure and private ML, we performed an *exploratory factor analysis* to estimate the latent construct of *awareness*. During the pilot study, participants described in their own words which risks of ML and defenses they were aware of. This yielded four attacks against ML security and privacy and 14 possible defenses. Following [86] which states that awareness corresponds to an individual's knowledge about risks, protection, and prevention methods, the resulting 18 items were expected to correlate with the concept of awareness. Therefore, we used them as a proxy for estimating awareness. For an overview on the factors see Table A.7 in Section A.2.2.3. We validated our factor analysis by using *Bartlett's test* of sphericity [16] ($\chi^2_{(2)} = 383.91$, $p < 0.001$). Additionally, we performed the *Kaiser-Meyer-Olkin test* [101, 102] ($KMO = 0.84$). Both criteria indicate the suitability of our factor analysis. The number of selected factors was determined based on a scree-plot highlighting that a solution based on one factor is sufficient by explaining 34% of variance in the data.

Calculating Awareness Scores. Furthermore, we performed a varimax rotation assuming that the factors are uncorrelated. We only considered items with loadings > 0.45 as correlating strongly enough with our factor. Thereby, five items were excluded from further analyses as shown in Table A.7 in Section A.2.2.3.

We used *Cronbach's Alpha* [51] to test the reliability of our constructed scale. With a value of $\alpha = 0.86$, the scale, consisting of 13 items, can be characterized as 'good' by statistic's definition.

Finally, to derive a single variable characterizing the level of a participant's awareness, we calculated factor scores. Note that these scores can only serve as an estimate of the unobservable concept of awareness, *i.e.*, the hypothetical value that a participant would exhibit if awareness was measurable. The resulting scores had a range of $[-1.55, 2.21]$ and were normalized to $[0, 1]$ for better interpretability. We refer to these normalized scores as *Awareness Scores*. Please note that due to the normalization, the endpoints of the scale, 0 (no awareness) to 1 (high awareness), are relative values rather than absolute ones. This is because normalization was performed by the factor scores of the survey participants. As a consequence, even if an individual exhibits a score of 1, this does not mean that this individual has perfect awareness of secure and private ML. Instead, it indicates that the individual has the highest awareness among all participants concerning the concept of awareness as defined for this study.

3.3.5 Limitations and Biases

Finally, we present possible limitations and biases of this work.

Note that studying ML practitioners is a challenging task due to the difficulty of recruiting such a specialized population. As a consequence, our sample of participants contains some demographic biases. The first bias is a sampling bias regarding geographic location. Even though the great effort was taken to reach ML practitioners from all over the world, the majority of our participants indicated currently working in Europe. The second bias results from the application fields in ML among our participants. See Table A.3 in Section A.2.2.2 for a detailed overview on occupation fields. Another imbalance concerns the years that our participants indicated working with ML. The largest groups of participants indicated working with ML for 1-3 years (49, 59%). As mentioned above, this observation is congruent to the Kaggle survey (55%) [98]. Hence, it does not necessarily reflect a bias. Instead, the observation can be caused by the current developments in the field of ML and the large number of ML-related positions being created in recent years. Yet, as a consequence, our results might not be representative of all ML practitioners in all fields and at all levels of experience.

Furthermore, our difficulties in recruitment were aggravated by the fact that we did not offer monetary compensation for participation in our study. Note that the decision to not offer such compensation was a deliberate one with the aim of preventing dishonest participants who just click through the study to obtain their reward. Hence, our ML practitioners' decision to participate was voluntary and, therefore, depends to a large degree on individual motivation and interest in the topic. Based on the pros and cons, we decided to accept the risk of sampling too many ML practitioners who are already more interested in the topic.

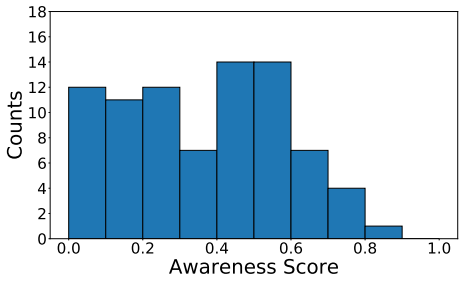


Figure 3.1.: **Awareness Scores.** Count plot of the participants' normalized awareness scores. 0 indicates no awareness and 1 high awareness. Figure adapted from [29].

As a consequence of the above-mentioned limitations, we are aware that the conclusions based on this survey might not generalize to all ML practitioners. Instead, this work can be considered as a starting point to assess ML practitioners' awareness and the state of affairs in secure and private ML. This can inform the improvement of existing and the design of new and better tools and standards in the field. Also, note that in particular the qualitative results from the study are independent of the sample size and provide valuable insights into the field.

3.4 Results

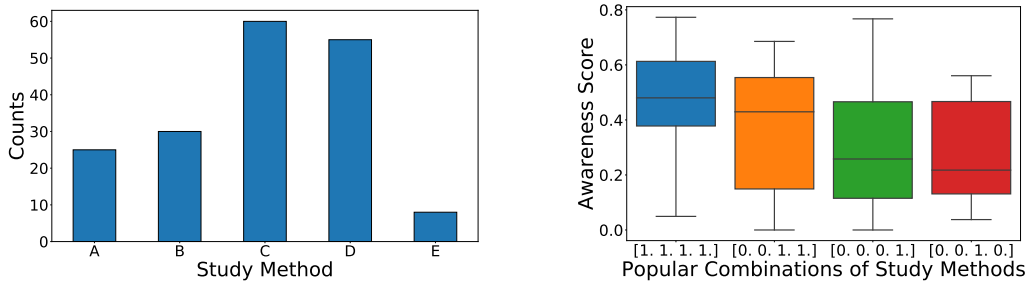
The four main findings of our study can be summarized as follows:

- The average awareness of security and privacy threats and protection measures among the surveyed ML practitioners is comparatively low.
- Academic education seems to have no significant impact on awareness of secure and private ML.
- ML protection methods put into place, especially for improving privacy, are less well-known than traditional and ML-specific security measures.
- The introduction of the GDPR appears to have no far-reaching impact on ML workflows in particular, and leaves the studied ML practitioners with several uncertainties.

This summary is taken from [29]. In the following, we depict the results more in detail to answer our two research questions.

3.4.1 RQ1: Machine Learning Practitioners' Awareness

When evaluating the practitioners' individual perceptions of the importance of securing their ML models, we found that 54 participants (65%) consider this task 'important' or even 'very important'. However, at the same time, our results suggest that our surveyed participants exhibit, in general, relatively low awareness



(a) Study methods used by ML practitioners to build awareness. A: 'University', B: 'Workshops', C: 'Practice', D: 'Self-Study', E: 'Other'.

(b) Distribution of ML practitioners' awareness scores for the four most frequently mentioned combinations of learning methods used by the participants.

Figure 3.2.: **Study Methods for Secure and Private ML.** The binary arrays in (b) decode whether the following methods were applied (1) or not (0): [University, Workshops, Practice, Self-Study]. The box plots in (b) correspond, from left to right, to 15, 14, 15, and 13 mentions respectively. Figures adapted from [29].

regarding security and privacy in ML. Figure 3.1 depicts the awareness scores over all survey participants. The quantiles of the normalized awareness score can be reported as $q_{0.25} = .173$, $q_{0.5} = .389$, $q_{0.75} = .522$, with scores between 0 (no awareness) to 1 (high awareness). In particular, our results do not show a correlation between the ML practitioners' perceived importance of securing their ML models and their respective awareness ($r_{(81)} = .073$, $p^* = .62$). We, therefore, suspect that our results regarding the practitioners' perceived importance might contain a desirability bias, based on the survey's introductory text mentioning researching security and privacy awareness as the goal of our study.

Building Awareness. We tested the hypothesis that an ML practitioner's education level has no influence on their awareness in the field. Our results suggest that, indeed, the education is no suitable indicator for awareness ($\chi^2_{(4)} = 0.89$, $p^* = .827$). We hypothesize that this is due to universities having only recently started to extend their study programs around artificial intelligence and ML [19]. This would explain why the impact of education in the field does not yet translate into the working environments, and hence, to our study participants.

Furthermore, secure and private ML are very specific sub-fields of ML, and therefore, they might not be taught extensively in universities. This hypothesis is backed up by our results suggesting that only 25 (30%) of our ML practitioners specified having acquired their knowledge around secure and private ML in university. Most of our participants stated that they have built their awareness through 'practice' (61, 73.5%) and 'self-study' (56, 67.5%). These findings are congruent with the findings

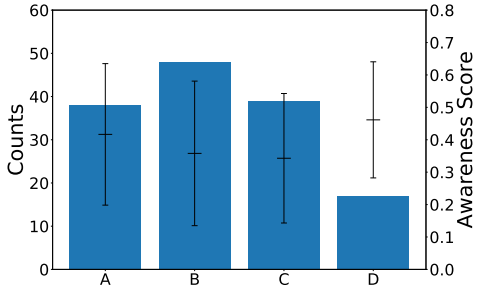


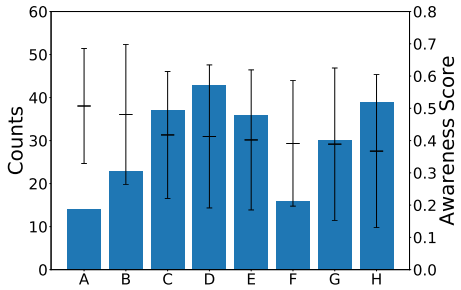
Figure 3.3.: **ML Practitioners' Working Environment.** Number of participants per working environment and the average awareness value with its standard deviation. A: 'Industry', B: 'Academic Research', C: 'Industrial Research', D: 'Hobby'. The bars do not add up to the sample size of 83 due to the possibility of giving multiple answers. Figure adapted from [29].

of Balebako *et al.* [12]. Figure 3.2a presents a compact overview on our participants' study methods for building awareness. Note that the question on methods used to build awareness allowed for multiple answers, asking the participants to select all applicable options. Therefore, the total count in the figure does not correspond to the number of participants $n = 83$. Due to the possibility for the participants to specify more than one study method, our results do not allow us to individually assess which study method has the largest influence on the ML practitioners' awareness. However, the results support the general assumption that the more sources for learning an individual relies on, the higher their respective awareness. Figure 3.2b shows the four most frequently reported combinations of learning methods and the respective awareness scores.

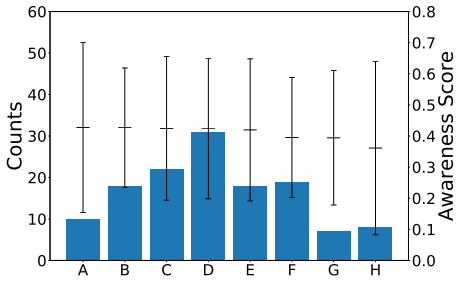
Apart from their answers to the closed questions in our pilot and full study, several participants also used free-text fields to add comments, such as *"I didn't yet work with sensitive data, so I didn't yet have to build that knowledge"*, or *"I never thought about securing my machine learning systems, mainly because they are research oriented rather than production oriented systems"*. These answers motivated us to study the ML practitioners' working environment to gain insights into what factors contribute to determining their awareness.

Working Environment. Our strongest finding when it comes to the impact of their work on the practitioners' awareness highlights that the number of years that an individual works with ML, professionally or as a hobby, has the largest effect on their individual levels of awareness ($r_{(81)} = .36$, $p^* = .005$).

We furthermore investigated whether the concrete environment where an individual works with ML has an impact on their awareness. Therefore, we considered 'industry', 'industrial research', 'academia', and 'hobby' as possible working envi-



(a) Data Type. A: 'Audio', B: 'Location', C: 'Tabular', D: 'Images', E: 'Text', F: 'Video', G: 'Metadata', H: 'Sensor'.



(b) Data Domain. A: 'Financial', B: 'Smart Env. and IOT', C: 'Public Security', D: 'Customers and Users', E: 'Transport and Traffic', F: 'Social Media', G: 'Medical and Health', H: 'Weather and Environment'.

Figure 3.4.: **Data vs. Awareness.** Influence of data on ML security and privacy awareness with the respective average awareness score and its standard deviation. Figures adapted from [29].

ronments. Since these environments differ significantly regarding their workflows and goals, a difference in the individuals' awareness could be expected. Due to the possibility of an individual working with ML in several of these environments at the same time, we asked our survey participants to select all options that apply to them. As a consequence, neither group-wise comparisons nor Friedman tests are applicable for analyses. Therefore, we visually depict in Figure 3.3 the absolute numbers of participants who indicated to work in a specific environment together with their average normalized awareness scores. The figure indicates that individuals working in the industry have a slightly higher awareness than individuals working in other environments. Yet, the differences are not significant. This result is not surprising since the fact that individuals can work in several environments at the same time can cause spillover effects.

Building on Pieczu *et al.* [149], we investigated whether the size of an organization in terms of employees that an ML practitioner works in has an impact on their security and privacy awareness. Such awareness can, for example, be built, through organizational security policies and guidelines. However, our results do not show a correlation between the size of their organization and the ML practitioners' awareness ($\chi^2_{(8)} = 15.26$, $p^* = .109$).

Data. As another aspect of an individual's working environment, we studied the impact of the data that ML practitioners work with on their individual awareness. For example, the type of data might have an impact on awareness. This is because, *e.g.*, individuals who work with medical patient data or portrait images might have a higher awareness than individuals who work with industrial sensor or with

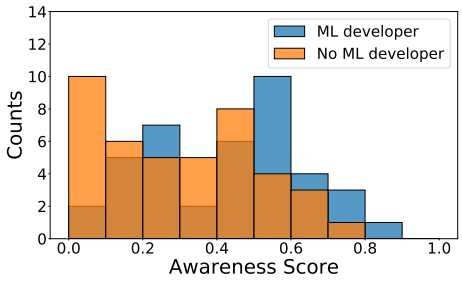


Figure 3.5.: **Awareness among (Non-)Developers.** Distribution of the ML practitioners' normalized awareness score grouped by whether they are considered to be ML developers or not. Figure adapted from [29].

weather data. Figure 3.4a provides an overview on the eight groups of different data types (image, video, audio, text, location, meta, sensor, and tabular data) and the respective practitioners' average awareness. The figure shows only slight differences across the different data types. Note that some data types, such as audio (.50) and location data (.48), lead to a higher score than, for example, images (.41). This might be explained by the fact that this data is particularly sensitive, and thereby, has a positive influence on the respective ML practitioners' awareness. However, this could be the same for video or metadata, which, in our study, led to the lowest awareness score (.39 and .39) apart from sensor data. Figure 3.4b shows a similar analysis regarding the domain of the data that our survey participants work with. It can be seen that there are only slight differences in awareness between awareness of ML practitioners depending on the domain of data that they work with.

The small differences in average awareness between different data types and data domains can be caused by the fact that, most likely, practitioners in their working environment, have to deal with different kinds of data, which can cause spillover effects in our results again. Additionally, it can be due to the fact that all kinds of data can contain both sensitive and non-sensitive information. Therefore, we also explicitly surveyed whether an ML practitioner was working with sensitive data, *i.e.*, data that is directly or indirectly related to human beings. It might be expected that individuals working with data that is directly related to human beings exhibit the highest awareness. However, our results show the highest average awareness among ML practitioners who work with data that is only indirectly related to individual human beings (.44). The average awareness of ML practitioners working with data that is directly related to individuals or not related to them at all is lower (.35 and .32, respectively). A Kruskal-Wallis test did not discover significant differences among individuals from the three groups of data ($\chi^2_{(2)} = 4.09$, $p^* = .194$). This indicates that even working with data that is directly related to individual human beings has no significant effect on our ML practitioners' awareness.

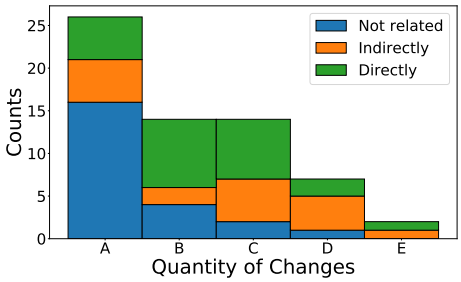


Figure 3.6.: **GDPR and Workflow Changes**. Changes in ML related workflows due to the introduction of the GDPR are grouped by whether the data that the ML practitioners are working with is directly related to individuals, indirectly or not related at all. Degree of changes: A: 'Not at all', B: 'Very little', C: 'Somewhat', D: 'Very much', E: 'To a great extent'. Figure adapted from [29].

Tasks. Moreover, we studied if ML practitioners' daily tasks have an impact on their awareness of security and privacy in ML. In particular, we distinguish between individuals who perform high-level tasks, such as applying ML libraries, and the ones who perform more low-level tasks, such as developing libraries. As introduced in Section 3.3.3, we created the group of 'ML developers' post hoc, based on the participants' daily tasks. Our analysis shows that ML practitioners who belong to the group of ML developers have a significantly higher average awareness than non-developers ($U_{(41,42)} = 584.0$, $p^* = .018$). We visualize these results in Figure 3.5. The observed effect can, most likely, be explained by the fact that developers require a deeper understanding of ML than individuals performing more high-level tasks.

GDPR. As the last factor that can potentially influence ML practitioners' awareness of security and privacy, we considered legal requirements at the example of the GDPR. Since the GDPR might not apply to study participants from outside the European Union, we excluded these participants from the analysis. Therefore, the following results reported regarding the GDPR only refer to 64 ML practitioners who indicated working within the European Union.

When studying what group of ML practitioners experienced the largest changes in their ML-related workflows due to the introduction of the GDPR, we relied on their self-assessment. Figure 3.6 presents this self-assessment, grouped by whether the data an ML practitioner works with is directly related to individuals, indirectly or not related. Surprisingly, the majority of ML practitioners in our study reported that they did not experience any changes in their workflow. Unsurprisingly, among these individuals, most reported dealing with data that is not person-related. However, the opposite effect could not be confirmed through the data: individuals working with data that is directly related to human-beings did not report significant changes in their workflows, either. Instead, their reported changes are similar to the changes

reported by the ML practitioners who work with data that is indirectly related to individuals. We assume that this similarity is due to the GDPR also requiring data that is only indirectly related to persons to be protected in order to prevent re-identification. An additional Kruskal-Wallis test ($\chi^2_{(2)} = 12.39$, $p^* = .005$) and a pairwise comparisons between the different degrees of data-sensitivity support the findings visualized in Figure 3.6. Indeed, the group of ML practitioners working with non-human related data differs significantly from the other two groups when it comes to changes in the ML-related workflows ($U_{(23,17)} = 97.0, p^* = .005$), ($U_{(23,23)} = 130.0, p^* = .003$). No statistically significant changes can be observed between the two groups of individuals working with indirectly or directly human related data ($U_{(17,23)} = 171.5, p^* = .38$).

In addition to the closed questions, we also included several free-text questions concerning the introduction of the GDPR in our questionnaire. The participants' answers highlighted that changes implemented after the introduction of the GDPR mainly affect the general workflows and not the specific ML-related ones. The changes that were reported include the reduced collection of data, data handling (*e.g.* anonymization and documentation), and data storage (*e.g.* location and security measures of the servers, access control, and encryption). One participant, in particular, stated *"It [the GDPR] is more important when it comes to data collection than the training phase or ML model development"*. The few changes that were reported specifically for the ML-related workflows include, according to, for example, one participant's statement, *"I can not [sic] use a lot of features that I used before and so, the strategy to solve some problems has changed"*. However, several ML practitioners stated that they were insecure when it comes to the changes that the GDPR requires for their ML workflows. These insecurities included, above all, the *Right to be Forgotten*.[1] Most practitioners stated that they lack knowledge on the implementation of this requirement, as can be illustrated by the statement of one participant who specified: *"Since GDPR requires the "right to be forgotten", we would need to artificially keep a correlation in order to be able to forget a specific individual, which, however, violates the GDPRs requirement to only store information that is required to operate a system"*.

As the last aspect concerning GDPR-related changes in ML workflows, it seems that third-party infrastructure and service providers have a lot of influence on the security and privacy of ML. Several participants stated in our survey that they rely on these parties without questioning them. For example, one participant stated that to implement the GDPR-related changes in their ML workflows, they *"checked some boxes in AWS"*. Another participant explained more in detail: *"I heavily rely on third party services to train and store my data, such as google cloud platform and AWS. Thereby, I hope their default security support is enough to protect my models and data"*.

[1] The Right to be Forgotten (Article 17, GDPR [188]) states: "The data subject shall have the right to obtain from the controller the erasure of personal data concerning him or her without undue delay and the controller shall have the obligation to erase personal data without undue delay"

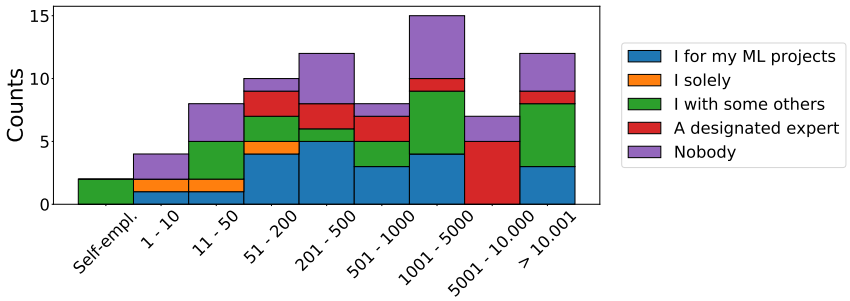


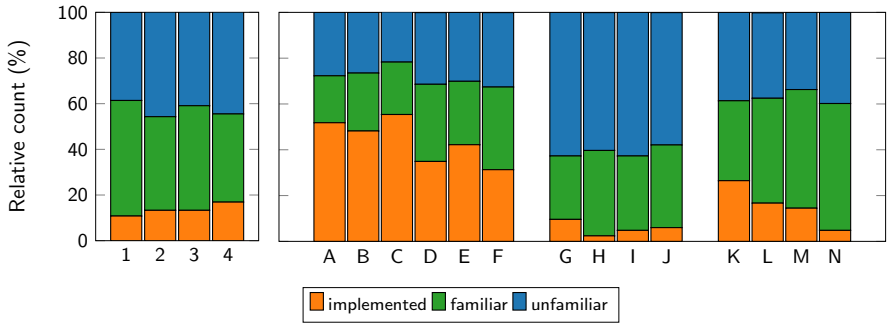
Figure 3.7.: **Distribution of Responsibility for Securing the ML Models.** (*Who takes care of the security of the machine learning (ML) models in your working environment?*) by company size. Figure adapted from [29].

3.4.2 RQ2: State in Secure and Private Machine Learning

After studying the individual ML practitioners' awareness, we turned to answer RQ2 by assessing the state of secure and private ML in practice.

Responsibilities. As a first insight into the state of secure and private ML in practice, we investigated who in the ML practitioners' working environment carries the responsibility to secure the ML models. The results are visualized in Figure 3.7. We could not reject the null hypothesis that the question of who is responsible for ML security is stochastically independent of the organization size ($\chi^2_{(32)} = 43.11$, $p^* = .182$). It seems that independent of the size of the organization, most of the time, the ML models' security is taken care of—either by an individual, a collective, or a designated expert. However, across organizations of all sizes, there seem to be cases where nobody is responsible for ML security. Through our participants' additional free-text answers regarding responsibilities for ML security, it, furthermore, seems that, in some cases, ML security is taken care of by a process implemented by the organization, or by each individual in charge of model deployment. Several participants also specified that their organization's IT department or security or data science teams implement ML security. Our results also indicate that there is no correlation between an individual's educational degree and their individual responsibility for securing their organization's ML models ($U_{(47,35)} = 774.5$, $p^* = .395$). The same holds for the years that a participant is already working in ML ($U_{(48,35)} = 728.5, p^* = .212$). This finding is surprising since our results reported in the previous section show a significant positive correlation between the years that an individual works with ML and their awareness. Therefore, one might have expected individuals with longer working experience to be in charge of securing the ML models.

Attacks. Moreover, we investigated the ML practitioners' awareness of attacks and threats against their ML models. Our findings highlight that 24% of our participants have never heard of any of the four attacks mentioned in our questionnaire



(a) Attacks. (b) Defenses.

Figure 3.8.: **ML Practitioners' Familiarity with Attacks and Defenses.** Overview of the participants' awareness of the attack and defense methods mentioned in the questionnaire. For *attacks*, the question was if the participants had already implemented measures to defend against it, or if they were familiar, or unfamiliar with it. 1: Inversion Attacks, 2: Impersonation Attacks, 3: Poisoning Attacks, 4: Evasion Attacks. For *defenses*, the participants were asked if they had implemented this defense, or if they were familiar, or unfamiliar with it. Cluster 1 (A: Data Sanitization [136], B: Access Control [163], C: System Security [172], D: Ensemble Learning [55], E: Data Provenance [36], F: Observing Model Input at Inference Time [15]), Cluster 2 (G: Differential Privacy [58], H: Homomorphic Encryption [79], I: Watermarking [131], J: Privacy Preserving Record Linkage [186]), Cluster 3 (K: Smoothing [50], L: Introducing Delay [167], M: Adversarial Training [78], N: Federated Learning [105]). Figure adapted from [29].

(inversion, impersonation, poisoning and evasion attack). This aligns with findings reported by Kumar *et al*. [108]. The participants familiarity across the four attacks was rather uniformly distributed and did not exhibit significant differences ($\chi^2_{(3)} = 1.12$, $p^* = .772$) as also depicted in Figure 3.8a. More than half of the surveyed participants have at least heard of the attacks. However, only 10-20% of them have actually implemented measures to defend against the respective attacks.

Defenses. When it comes to defense methods to protect ML models' security and privacy, the ML practitioners' familiarity is less equally distributed. In fact, visual and statistical evaluation of the familiarity with defense methods highlights significant differences over different defense methods ($\chi^2_{(13)} = 93.19$, $p^* < .001$). Figure 3.8b depicts the defense methods clustered according to the participants' familiarity with them. The clustering was obtained using the K-Means algorithm [96], while the number of clusters was obtained by the elbow method [104]. This resulted in the generation of three clusters that, interestingly, map to semantic units.

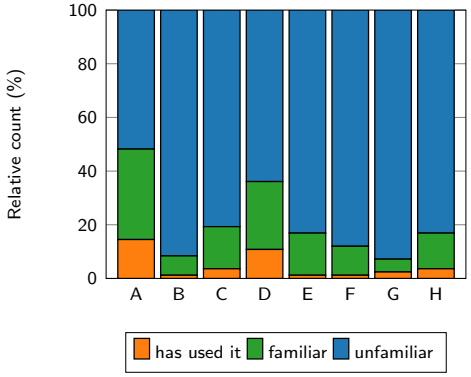


Figure 3.9.: **ML Practitioners' Familiarity with Libraries.** Libraries to support security and privacy in workflows: A: TensorFlow Privacy [122], B: Cleverhans [139], C: PySyft [159], D: Googles Differential-Privacy [195], E: Uber SQl Differential-Privacy [97], F: AdverTorch [56], G: Foolbox [153], H: ART Toolbox [135]. Figure adapted from [29].

The first cluster contains methods that up to half of the ML practitioners reported having implemented already (*data sanitization* [136], *access control* [163], *system security* [172], *ensemble learning* [55], *data provenance* [36], and *observing model input at inference time* [15]). All these methods implement rather classic than specific ML security, which can serve as an explanation of why these defenses are so prevalent.

In the second cluster, we find methods used for privacy and confidentiality protection of ML models (*differential privacy* [58], *homomorphic encryption* [79], *watermarking* [131], and *privacy preserving record linkage* [186]). For most of these methods, our study participants indicated being unfamiliar with them. However, only a minority of ML practitioners have already implemented them.

The third cluster contains methods that can be applied to ensure security of ML models (*smoothing* [50], *introducing a delay for model interaction* [167], and *adversarial training* [78]), or to protect the training data (*federated learning* [120]). Similar to the second cluster, most participants have not implemented these methods. However, the participants' theoretical familiarity with these methods seems to be higher than in the second cluster.

Libraries. Apart from their familiarity with attacks and defenses, we also surveyed ML practitioners concerning their familiarity with selected libraries to support the implementation of security and privacy in ML. As stated in Section 3.3.1, we selected the libraries according to the participants' answers in the pilot study. All the resulting libraries are Python libraries which can be explained by the popularity of this language in building ML applications [189]. Additionally, also Kumar *et al*. [108] reported that 16 of 28 organizations included in their survey use Python

frameworks, *e.g.* Keras, TensorFlow, or PyTorch while ten rely on ML-as-a-service providers, and only two build their ML systems from scratch. In Figure 3.9, we display our participants' familiarity with the given libraries. The figure highlights that across all libraries, the percentage of ML practitioners who have actually used them is comparably low. The most popular library seems to be TensorFlow Privacy with roughly 50% of all practitioners being familiar with it.

3.5 Discussion of the Findings and Outlook

This section discusses the findings of our study and gives an outlook on further research required in the area, structured according to our two research questions.

3.5.1 *RQ1:* How is ML security and privacy awareness built, and which conditions contribute to the degree of knowledge with respect to threats and corresponding defenses?

Concerning the first research question RQ1, our study yielded three main results.

1) The surveyed ML practitioners' awareness of attacks or threats, as well as of security and privacy practices in ML is relatively low. Similar findings were reported by Kumar *et al.* [108] for ML and by the studies presented in Section 3.2 for general security and privacy practices among developers. The finding might be explained through the fact that in both ML and standard IT applications, the security and privacy requirements are usually kept separate from the requirements that concern functionally [61]. Additionally, functionality is usually taken care of before implementing the security and privacy requirements [132]. This might lead to these being fully neglected when deadlines approach. Another possible explanation could be given by the fact that many companies nowadays are still in the early stages of adopting ML. Therefore, security and privacy might not yet be driving factors. Instead, the products are still rather experimental and proofs of concepts [113].

2) Most ML practitioners did not receive academic training on ML security and privacy. Similar to the results reported for general security practices [12], our study highlights that most of ML practitioners did not receive any formal academic training regarding ML security and privacy. As shown in Figure 3.2b on page 33 in the previous section, only roughly one third of our participants reported having learned about ML security and privacy in university. This result is not surprising given that ML security and privacy is an even more specific field than general security and privacy, and that only in recent years, there has been a significant increase in ML-related programs in universities. The latter might, in the upcoming years, lead to an increase in the overall literacy on the topic. However, the results of this study also suggest that formal education degrees do not significantly correlate with practitioners' awareness of ML security and privacy. Instead, the years of

work experience in ML exhibit a significant positive correlation with individual awareness. Similar results were also reported by Acar *et al.* [4]. This suggests that improving academic education on the topics might just be one possible step towards raising ML practitioners' awareness. This intuition is supported by the finding of our study showing a positive correlation between the extent of different educational resources used and awareness of security and privacy in ML.

3) Our study showed no correlation between the size of the organization where the surveyed ML practitioners work in and their personal awareness. In general, one might expect ML practitioners in larger organizations to exhibit a higher awareness. This is due to such larger organizations often implementing organization-wide coding and security guidelines, peer-review processes, and internal training. However, it is also possible that awareness in smaller organizations, such as start-ups, is at a similar level because the ML practitioners there need to deal with security and privacy by themselves whereas practitioners in larger organizations might have dedicated experts whom they can rely on to implement them, instead.

3.5.2 *RQ2:* What is the current state of affairs concerning ML security and privacy among ML practitioners?

We report four main findings concerning the state of affairs in ML security and privacy in practice.

1) The participants' familiarity with protection strategies for their ML models is very unevenly distributed. Interestingly, we observed that the clusters regarding familiarity with the defense methods correspond well to semantic units. The cluster that exhibits the highest familiarity contains methods from general security. Defenses from the cluster containing methods for (partly) ML-related security are less well known and implemented. The cluster with the lowest familiarity among the ML practitioners contains methods for protecting ML privacy and confidentiality. This suggests that, so far, these methods have received the least attention. However, the finding leaves open the question if this leads to privacy in ML models being less protected than security. Alternatively, one could assume that privacy is mainly considered in other parts of the workflows around the ML models as suggested by several participants' free-text answers.

2) Over all organization sizes, there are cases where nobody is responsible for the model security and privacy. Independent of the practitioners' awareness, within organizations of all sizes, there exist cases where nobody is in charge to ensure the ML models' security and privacy. This finding is aligned with results reported by Flechais *et al.* [61] who conducted a case study in the area of application development. They showed that often, no developer was explicitly responsible for the security of the project.

3) The introduction of the GDPR had comparatively little impact on ML workflows in particular. In general, when it comes to the GDPR, several of our survey participants mentioned uncertainties about the formulation of the GDPR and its technical applicability for ML in particular. By inviting more ML practitioners to join in the policy-making processes around data protection, such issues could be mitigated in the future [11]. This could be complemented by the provision of precise guidelines on how to implement the regulations for ML, similar to guidelines that exist already, for example, in the area of secure app development [10, 38]. However, to date, this represents a difficult task since research on secure and private ML is still at a stage where concrete recommendations are only possible for some well-defined issues and not in general.

4) The surveyed practitioners' familiarity with the presented security and privacy libraries for ML is low. Integration of security and privacy can be facilitated by the use of the right tools and libraries. Research by Wurster and van Oorschot [196] suggests that tools that include security should be more usable than tools that do not in order to encourage developers to use them. This suggestion is also formulated by other research work [94, 174]. In a similar vein, the low popularity of libraries supporting secure and private ML might be explained by their limited usability, their applicability to very specific scenarios, and their lack of expressive documentation. As a consequence, to increase the application of libraries for secure and private ML, in the future, it might be valuable to investigate ML practitioners' working practices further beyond the scope of this work, study their functional needs thoroughly, and adapt existing libraries according to these findings [170]. In the adaptation of the libraries and tools, also human factors should be taken into account, such as the practitioners' expectations of the libraries' functionalities, behaviors, and interfaces [40]. Moreover, an improvement of existing documentation and the provision of secure and private code examples could be helpful [108, 130, 149]. The latter ones are shown to improve security and privacy in general-purpose code since developers follow such examples very closely [94].

As an additional factor, the inherent support and integration of security and privacy protections into third-party software should be implemented. Our survey's results, similar to the results of Kumar *et al*. [108], highlight that ML practitioners rely on third-party services heavily for both functionality but also for the integration of security and privacy. To prove the third-parties' correct integration of the methods, some certificates could be issued by a trusted certifying authority. Such assessments are already widely available for different software products, *e.g*. cryptographic modules, database servers, or operation systems [35]. We assume that by employing adequate libraries, tools, and third-party software, and by relying on automated detection of attacks and application of defenses, ML practitioners can be supported in implementing ML security and privacy. The question of individual awareness on the topic would then shift to awareness on choosing the right tools and employing them correctly rather than being aware of each individual threat and defense.

3.6 Conclusion

Our study investigated awareness of security and privacy among ML practitioners, their practices with regard to the field, and the impact of the introduction of the GDPR on their workflows. Our results highlight that our survey participants' awareness of ML security and privacy in general, and ML-specific privacy measures in particular, is comparably low. Furthermore, we observed a low familiarity with existing libraries to support secure and private ML development. Finally, third-party services seem to play an important role in ensuring ML security and privacy since ML practitioners trust them to provide sufficient protection.

Chapter 4

Data Reconstruction Attacks in Federated Learning

The previous chapter highlighted the important role of third-party services and their providers in implementing privacy protection in Machine Learning (ML). One service or protocol that was, for a long time, promoted as a solution for implementing privacy-preserving ML is Federated Learning (FL). As depicted in more in detail in Section 2.2, FL allows to use the private data of its clients to jointly train an ML model without this data ever having to leave the clients' devices. Instead, each device computes model updates and sends them to a server which aggregates the updates to produce a shared model. If we were to *assume* that the model updates do not reveal the clients' data, FL would, indeed, preserve a notion of privacy. However, this assumption has already been contested by prior work several times: model updates sent to the server do not only leak training data membership [125], and thereby allow an attacker to tell if a given data point was used in training. At the same time, they reveal properties of the training data [68, 125] and even allow to partially *reconstruct* clients' training data [70, 193, 200, 205, 207]. Such data reconstruction represents the most severe form of privacy disclosure for the clients in an FL protocol.

In this chapter, we present *adversarial weight initialization* as a novel attack vector against privacy in FL. The idea of adversarially initializing model weights was, so far, solely explored for producing ML models with decreased training performance [81]. We introduce two types of adversarial weight initializations to attack privacy. The first one allows to extract client-data directly from model updates, *i.e.*, the gradients sent from clients to the server. The second one enables to forward clients' input data unaltered over fully-connected and convolutional model layers and, thereby, allows to extend our data extraction attack to a broader range of model architectures. The attack vector of adversarial weight initialization naturally integrates in the FL protocol where the server holds full control over the shared model. See Figure 4.1 for an overview on the course of our novel active data extraction attack.

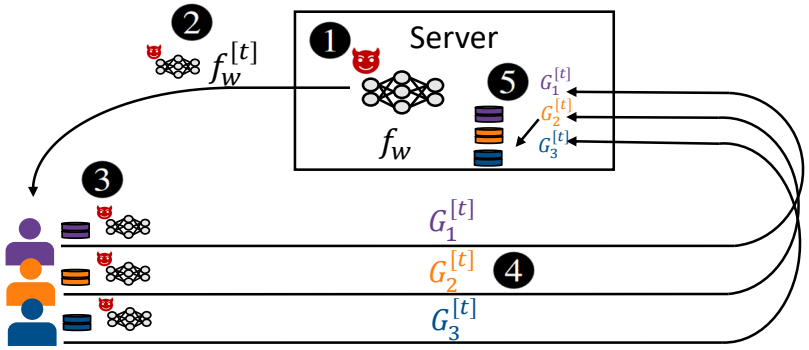


Figure 4.1.: **Course of our Novel Active Data Extraction Attack.** ❶ The server adversarially initializes the weights of the shared model. ❷ The server sends the manipulated model out to the clients. ❸ Clients calculate model gradients on their local private data and ❹ send these gradients to the server. ❺ The server then extracts the private client-data directly and perfectly, *i.e.* with zero error, from the received gradients.

In the following, in Section 4.1, we first present prior data reconstruction attacks against FL and highlight their limitations. In Section 4.2, we go a step beyond these attacks and introduce our novel observation that gradients sent from clients to the server include full, memorized training data points. Afterwards, in Section 4.3, we propose an adversarial weight initialization that builds on this observation and can significantly increase the proportion of perfectly reconstructable client data points. Thereby, this adversarial weight initialization can be used to perform a data extraction attack in FL. Remember from Section 2.3 that data extraction is a specific form of data reconstruction where data can be extracted directly and without any error from the ML model. We then present our second type of adversarial weight initialization in Section 4.4 which allows to forward ML models' input data unaltered over fully-connected and convolutional model layers. Thereby, it makes the data extraction attack applicable to a broader range of ML model architectures Furthermore, we propose methods on making the manipulations of the model weights inconspicuous. In Section 4.5, we experimentally evaluate our novel adversarial weight initializations. We show that our data reconstruction attack is even effective when the model gradients are computed over large mini-batches of complex data, even when this data belongs to the same class—scenarios in which previous attacks usually fail to obtain high-fidelity reconstructions [190]. Additionally, we depict that while previous reconstruction attacks rely on solving complex optimization problems that require tens of thousands of back-propagation iterations over the model, our attack extracts individual training inputs by simply projecting the appropriate portions of the clients' gradients onto the input domain. We thereby reduce the computational time needed to restore individual training data points from several hours to a few milliseconds. Finally, in Section 4.6, we conclude with

a brief summary of our findings and give an outlook on the implications of our work for privacy in FL.

A preprint with the results presented in this chapter has been made available online [30] prior to composing this dissertation. Therefore, main parts of this chapter contain content directly taken from [30].

4.1 Related Work on Data Reconstruction Attacks

This section, adapted from [30], introduces prior work on data reconstruction attacks and discusses limitations of attacks that are based on iterative optimization.

4.1.1 Prior Data Reconstruction Attacks

Phong *et al*. [148] were the first to show how gradients leak information that can be used to recover training data from single neurons or linear layers. Recent work [70, 148, 193, 200, 205, 207] proposed that the server or clients involved in FL training can launch data reconstruction attacks based on either training a Generative Adversarial Network (GAN) [77] or solving a second order optimization problem.

Class-wise Representation Reconstruction Attacks. Hitaj *et al*. [90] were the first to propose a GAN-based data reconstruction attack, called DMU-GAN. The attacker must know the dataset's classes, and the reconstructed data points are generic representations of class-wise properties rather than individual client data points or classes. Wang *et al*. [193] suggested mGAN-AI, which extends DMU-GAN's reconstruction attack to per-client class-wise representations, but still does not extract individual data points. Additionally, both methods require access to data from the same distribution as the clients' training data. [176] observes that class-wise representations are embedded in model updates even without the need to reconstruct them using a GAN, and suggest defenses.

Optimization-based Instance Reconstruction Attacks. Several attacks aim to reconstruct individual client data points while also relaxing the assumption that data labels are available to the attacker. Zhu *et al*. [207] proposed Deep Leakage from Gradients (DLG), where a data reconstruction attack is formulated as a joint optimization problem on the labels and input data. In follow-up work, iDLG [205] sped up the convergence rate of DLG [207] by analytically computing the labels based on the clients' gradients of the last layer. These works, and other optimization-based ones [70], are limited to a setting where mini-batches only contain a single example, *i.e.*, $B = 1$. In subsequent work, GradInversion [200] regularizes DLG's objective to improve the extraction fidelity, attaining some success in extraction for mini-batches of size $B > 1$.

Algorithm 1: Optimization-Based Data Reconstruction [207]. Algorithm taken from [30].

Input: Gradients, $G_i^{[t]}$, received from victim client u_i at iteration t, Shared model $f_{\mathcal{W}}^{[t]}(\cdot)$ at iteration t.
Output: Reconstructed training data, $(\mathbf{x}_i^*, y_i^*)$

1: $(\hat{\mathbf{x}}^{[1]}, \hat{y}^{[1]}) \leftarrow (\mathcal{N}(0,1), \mathcal{N}(0,1))$ ▷ Initialize
2: **for** $\hat{t} \in [1, \hat{T}]$ **do**
3: $\quad \hat{G}^{[\hat{t}]} = \nabla_{\mathcal{W}} \mathcal{L}(f_{\mathcal{W}}^{[t]}(\hat{\mathbf{x}}^{\hat{t}}), \hat{y}^{\hat{t}})$ ▷ Dummy gradients
4: $\quad D^{[\hat{t}]} = \|G_i^{[t]} - \hat{G}^{[\hat{t}]}\|^2$ ▷ Dummy vs client gradients
5: $\quad \hat{\mathbf{x}}^{[\hat{t}+1]} \leftarrow \hat{\mathbf{x}}^{[\hat{t}]} - \alpha \nabla_{\hat{\mathbf{x}}^{[\hat{t}]}} D^{[\hat{t}]}$,
6: $\quad \hat{y}^{[\hat{t}+1]} \leftarrow \hat{y}^{[\hat{t}]} - \alpha \nabla_{\hat{y}^{[\hat{t}]}} D^{[\hat{t}]}$
7: **end for**
8: $(\mathbf{x}_i^*, y_i^*) \leftarrow (\hat{\mathbf{x}}^{[\hat{T}+1]}, \hat{y}^{[\hat{T}+1]})$

Data Extraction Attacks. As presented in Section 2.3.3, data extraction attacks can be considered a specific form of data reconstruction attacks where the individual training data points can be extracted directly and without any error from the ML model or its gradients. In concurrent work, Fowl *et al*. [63] consider a threat model in which the server also manipulates the shared model, similar to our setup. This attack relies on the existence of a fully-connected layer early within the network (otherwise, the attack adds it). Since this layer's weights have to contain many weight rows with the exact same weight values, this layer is inherently detectable.[1] Additionally, they do not discuss passive analytical-extraction attacks. Finally, our work generalizes their setup and performs successful extraction at arbitrary model layers, also for textual data.

4.1.2 Limitations of Optimization-Based Attacks

We hereby provide a brief exposition to Zhu *et al*. [207]'s DLG, as a representative case study of an optimization-based attack. Their approach, characteristic of optimization-based data reconstruction attacks, is given in Algorithm 1. It firstly randomly initializes a "dummy data point and corresponding label" $(\hat{\mathbf{x}}, \hat{y})$ and computes the resulting "dummy gradients" as $\hat{G} = \nabla_{\mathcal{W}^t} \mathcal{L}(f_{\mathcal{W}^t}(\hat{\mathbf{x}}), \hat{y})$. Then, they iteratively optimize the dummy data to produce gradients that are close to the original gradients $G_i^{[t]}$ by solving:

[1] This is inherent to the attack because the method relies on each row computing the exact same function on the data and binning its result by varying only the bias term, such that it becomes likely that a bin contains only one input. Conversely, our adversarial weights are initialized such that it is likely that an output neuron is only activated for a single input in a mini-batch while avoiding imposing a highly regular structure on the weight matrix.

$$\mathbf{x}_i^* = \arg\min_{\hat{\mathbf{x}}} \|G_i^{[t]} - \hat{G}\|^2, \quad y_i^* = \arg\min_{\hat{y}} \|G_i^{[t]} - \hat{G}\|^2. \tag{4.1}$$

DLG often fails to reconstruct high-fidelity data points and discover the ground-truth labels consistently because of a lack of convergence in the optimization. While other methods offer improvements (*e.g.*, iDLG [205] sped up the convergence by simplifying the objective in Equation (4.1) by first extracting the data labels from the gradients, and then optimizing only over the data points; and GradInversion [200] adds useful regularization), they suffer from the same pathology.

We identify several reasons for this. First, the gradient of the loss is non-injective *i.e.*, is not invertible everywhere: different mini-batches may yield nearly identical gradients [169]. This holds whether the client samples mini-batches that contain multiple data points or a single data point only, *i.e.*, $B = 1$. Second, optimization-based attacks converge to different minima due to the underlying randomness (see step 1 in Algorithm 1). These minima correspond to different possible reconstructions of the input that often differ from the original training points [200]. Third, optimization-based attacks are computationally expensive: they either need to train a GAN or solve a second-order gradient optimization problem. Instead, our attack extracts *exact* data points from the gradients without any optimization and without training a GAN.

4.2 Data Leakage from Model Gradients

This section contains a slight adaptation of our results presented in [30]. It highlights how and why the gradients of an Fully-Connected Neural Network (FC-NN) directly leak the individual training points they are computed on. In Section 4.2.1, we mathematically show that for a single training data point, *i.e.*, a mini-batch size of $B = 1$, perfect extraction from the network gradients is possible. Then, in Section 4.2.2, we motivate why it is also possible to perfectly extract a small number of individual data points from gradients, even when working with larger mini-batches of size $B > 1$.

4.2.1 Single-Input Gradients Directly Leak Input

It has been shown by Geiping *et al*. [70] that a single input data point $\mathbf{x}$ can be reconstructed from the gradients of any fully-connected layer which is preceded only by fully-connected layers and contains a bias $\mathbf{b}$. This holds if the gradient of the loss w.r.t. the layer's output $\mathbf{y} = \text{ReLU}(W\mathbf{x} + \mathbf{b}) = \max(0, W\mathbf{x} + \mathbf{b})$ contains at least one non-zero entry. For detailed proof of the above see Proposition D.1 in [70]. In particular, when considering the first model layer, reconstructing its input data *directly* corresponds to obtaining the original input data point $\mathbf{x}$. Let y_i denote the output of the i^{th} neuron of the first and fully-connected layer of a model, and let $\mathbf{w}_i^T$

be the corresponding row in the weight matrix and b_i the corresponding component in the bias vector. Assume $\mathbf{w}_i^T\mathbf{x} + b_i > 0$, and therefore, $\text{ReLU}(\mathbf{w}_i^T\mathbf{x} + b_i) = \mathbf{w}_i^T\mathbf{x} + b_i$. The reconstruction of the input $\mathbf{x}$ is done by calculating the gradients of the loss w.r.t. the bias and the weights as follows:

$$\frac{\partial \mathcal{L}}{\partial b_i} = \frac{\partial \mathcal{L}}{\partial y_i}\frac{\partial y_i}{\partial b_i} = \frac{\partial \mathcal{L}}{\partial y_i} \tag{4.2}$$

since $\frac{\partial y_i}{\partial b_i} = 1$, where $y_i = \mathbf{w}_i^T\mathbf{x} + b_i$. Furthermore,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i^T} = \frac{\partial \mathcal{L}}{\partial y_i}\frac{\partial y_i}{\partial \mathbf{w}_i^T} = \frac{\partial \mathcal{L}}{\partial b_i}\mathbf{x}^T. \tag{4.3}$$

Thus, if any $\frac{\partial \mathcal{L}}{\partial b_i} \neq 0$, perfect reconstruction is given by:

$$\mathbf{x}^T = (\frac{\partial \mathcal{L}}{\partial b_i})^{-1}\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i^T}. \tag{4.4}$$

According to Equation (4.3), the gradient of the loss w.r.t. the weights directly contains a scaled version of the input data. The exact scaling factor is $(\frac{\partial \mathcal{L}}{\partial b_i})$, which is the gradient of the loss w.r.t. the bias. This gradient is computed in the regular backward pass together with the gradient of the weights. Therefore, obtaining the scaling factor by just reading it from the gradients of the bias and inverting it to $(\frac{\partial \mathcal{L}}{\partial b_i})^{-1}$ comes at zero costs. The resulting factor can be directly applied to rescale the gradient of the weights and obtain the input data point $\mathbf{x}$, see Equation (4.4). Intuitively, the reason why there is a rescaled version of the input data in the gradients and why this would be beneficial for learning can be motivated by revisiting the simple perceptron algorithm [67]. When an input is misclassified, the weight update in the perceptron algorithm consists simply in adding this input to the weights, which makes the algorithm learn.

4.2.2 Mini-Batch Gradients Directly Leak Some Individual Inputs

It turns out that individual data point leakage is not limited to gradients computed over a mini-batch of size $B = 1$: we observe that gradients computed over larger mini-batches also sometimes leak individual training points. To forge an intuition for this phenomenon, Figure 4.2 visualizes the gradients of the first fully-connected layer's weight matrix of an FC-NN. We see that we are able to clearly distinguish some of the training data points within the rescaled gradients. This is despite the fact that these gradients were computed over a mini-batch of $B = 100$ inputs sampled from the CIFAR-10 dataset.

Figure 4.2.: **Passive Data Leakage from Model Gradients.** Data from the CIFAR-10 dataset, extracted from the gradients of the first 30 weight rows at the first fully-connected layer of a randomly initialized FC-NN [30].

Why do some gradients contain individual training data points? We denote a training data mini-batch by $X_b = \{x_1, x_2, \cdots, x_B\} \in \mathbb{R}^{(m \times B)}$ with $B > 1$. The gradient of this mini-batch X_b is equal to the average of all gradients computed for each of the data points $\{x_1, x_2, \cdots, x_B\}$ that make up the mini-batch. Let y_i denote again the output of the i^{th} neuron of the fully-connected layer, and let $\mathbf{w_i}$ and b_i be the corresponding row in the weight matrix and the component in the bias vector, respectively. Then the gradient $\mathbf{G}_{\mathbf{w}_i^T}$ and G_{b_i} of $\mathbf{w_i}$ and b_i can be computed as follows:

$$\begin{aligned} \mathbf{G}_{\mathbf{w}_i^T} &= \frac{1}{B} \sum_{j=1}^{B} \frac{\partial \mathcal{L}}{\partial y_{(i,j)}} \frac{\partial y_{(i,j)}}{\partial \mathbf{w}_i^T} \\ G_{b_i} &= \frac{1}{B} \sum_{j=1}^{B} \frac{\partial \mathcal{L}}{\partial y_{(i,j)}} \frac{\partial y_{(i,j)}}{\partial b_i} \end{aligned} \tag{4.5}$$

with $y_{(i,j)} = \text{ReLU}(\mathbf{w}_i^T \mathbf{x}_j + b_i)$. These equations illustrate that the gradient $\mathbf{G}_{\mathbf{w}_i^T}$ over the data mini-batch X contains a weighted overlay of all the input data points $\mathbf{x}_j$ from the mini-batch. The weighting, therein, depends on the contribution of each data point to the model loss $\mathcal{L}$.

We observe that, in some cases, all but one training data point $\mathbf{x}^*$ from the data mini-batch have zero gradients. This is due to the max operation in $\text{ReLU}(\mathbf{w}_i^T \mathbf{x} + b_i) := \max(\mathbf{w}_i^T \mathbf{x} + b_i, 0)$. When $\mathbf{w}_i^T \mathbf{x} + b_i$ is negative, the ReLU outputs zero, which results in zero gradients for the corresponding data point. When the gradients are zero for all data points but for the one data point $\mathbf{x}^*$, the weight gradient $\mathbf{G}_{\mathbf{w}_i^T}$ from Equation (4.5) becomes $\mathbf{G}_{\mathbf{w}_i^T} = \frac{1}{B} \frac{\partial \mathcal{L}}{\partial y_{(i,*)}} \frac{\partial y_{(i,*)}}{\partial \mathbf{w_i^T}}$ with $y_{(i,*)} = \text{ReLU}(\mathbf{w}_i^T \mathbf{x}^* + b_i)$. This reduces the data extraction from the case of $B > 1$ to the case of $B = 1$, for which we saw in Section 4.2.1 that the data point $\mathbf{x}^*$ can be perfectly extracted. In other words, $\mathbf{w}_i^T \mathbf{x} + b_i$ being negative for all data points but one results in accidental leakage of that data point—enabling its perfect reconstruction, *i.e.*, extraction by any passive attacker that can observe model gradients.

4.3 Adversarial Weight Initialization for Data Extraction

In this section, we show that the proportion of data points that can be perfectly extracted from the model gradients can be significantly increased by adversarially initializing the model weights. Note that for this type of attack, the attacker does not only require access to the model gradients but also needs the ability to manipulate the model weights. In FL, the server is in charge of the shared model, and therefore holds this exact ability which makes our data extraction attack very well suited for FL scenarios. The approach presented in this section is again taken from [30].

The previous section illustrates under which conditions model gradients leak data points to a passive attacker capable of observing these gradients. In the following, we show how an active attacker can amplify previously-accidental leakage during the passive attack by controlling the weights $\mathbf{w}_i$ and biases b_i. As we will elaborate more in detail in Section 4.5, while a passive attacker can extract roughly 20% of arbitrary data points from a batch size $B = 100$ for 1000 neurons (*i.e.*, weight rows in the fully-connected layer) on ImageNet, the active attack can more than double the number of extracted data points to 45%.

4.3.1 If-Else Logics over Relationships Between Data Features

Without loss of generality, we will suppress the bias term in the following considerations. The multiplication of a single weight row $\mathbf{w}_i$ corresponding to the i^{th} neuron at the fully-connected layer with some input data point $\mathbf{x}$ can be expressed as a weighted sum of all of the features in $\mathbf{x}$ as follows

$$\mathbf{y}_i = \mathbf{w}_i^T \mathbf{x} = \sum_{j=1}^{m} w_i^{(j)} x_j. \tag{4.6}$$

In $\mathbf{w}_i$, let $\mathtt{N}$ and $\mathtt{P}$ denote the sets of indices that hold the negative and positive weight components, respectively. Given ReLU activation, the i^{th} neuron is only activated on $\mathbf{x}$ if the sum of the features weighted by the negative components is smaller than the sum of the features weighted by the positive components:

$$\sum_{n \in \mathtt{N}} |w_i^{(n)} x_n| < \sum_{p \in \mathtt{P}} w_i^{(p)} x_p. \tag{4.7}$$

Therefore, $\mathbf{x}$ will yield non-zero gradients at the i^{th} neuron if and only if this particular relationship, expressed by Equation (4.7), between the features holds. When the inequality holds only for a single data point in a mini-batch, this data point can be individually extracted from the gradients, as described in Section 4.2.2. The idea behind our adversarial weight initialization is to set the components within each weight row corresponding to each neuron of the first fully-connected layer,

Algorithm 2: Adversarial Initialization of a Weight Row [30].

Input: Weight row $\mathbf{w_i}$ of length L, Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with mean μ and std σ, Scaling factor $s < 1$, Discrete uniform distribution $\mathcal{U}(\cdot, \cdot)$

Output: Adversarially initialised weight row $\mathbf{w_i}$

1: $\mathtt{N} = \{i | i \sim \mathcal{U}(1, L)\}$ where $|\mathtt{N}| = \frac{1}{2}L$ ▷ Select randomly indices for negative weights
2: $\mathtt{P} \leftarrow \{i \notin \mathtt{N} | i \in [L]\}$ ▷ Select indices for positive weights
3: $\mathbf{z}_- \sim \mathcal{N}(\mu, \sigma) | \mathbf{z}_- \in \mathbb{R}_-^{\frac{1}{2}L}$ ▷ Negative samples
4: $\mathbf{z}_+ = -s \cdot \mathbf{z}_-$ ▷ Positive samples
5: $\mathbf{w_i}[\mathtt{N}] \leftarrow \text{Shuffle}(\mathbf{z}_-)$ ▷ Initialize negative weights
6: $\mathbf{w_i}[\mathtt{P}] \leftarrow \text{Shuffle}(\mathbf{z}_+)$ ▷ Initialize positive weights

such that this relationship only holds relatively rarely in inputs, and is therefore likely to only hold for a single data point within a mini-batch.

4.3.2 Manipulating Weights for Data Extraction

Intuitively, our approach adversarially initializes each row of the weight matrix to increase the likelihood that *only one* data point in a given mini-batch will activate the neuron corresponding to that row. To achieve this, we initialize a randomly chosen half of the components of the weight row to negative values, and the other half to the corresponding positive values, by sampling from a Gaussian normal distribution. Negative weights are sampled with slightly larger absolute values than the positive weights to decrease the likelihood of multiple training points activating the corresponding neuron. See Algorithm 2 for a formalization of our adversarial weight initialization.

We use a relatively large standard deviation, such as $\sigma = 0.5$, so sampled points span widely across the weight range. This causes some features to be multiplied with relatively large positive or large negative values, and thereby increases the variability of the sums in Equation (4.7) across data points, ensuring that the inequality holds only for a few of them—in the best case only one—which allows for perfect extraction.

We use the scaling factor s to specify how much larger the absolute values of the negative weight components should be than the positive values. This determines how "aggressively" our activation causes weighted inputs to individual neurons to be negative, thereby to be filtered out by the ReLU function and to have zero gradients for most input data points. The ideal value of s when it comes to attack effectiveness is dataset dependent; it could be fine-tuned by an attacker through access to one mini-batch of data, or over time by evaluating the attack success on the gradients received from the users. However, to set s, the attacker does not rely on knowledge about the actual values of the input data features and how they

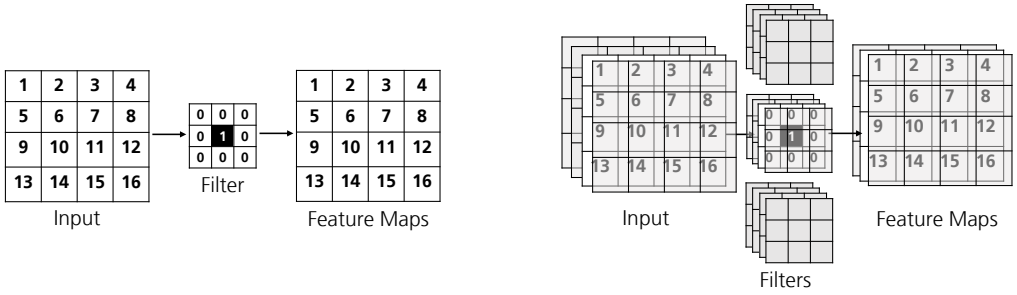
relate. In fact, the only assumptions we need to make is that our attacker knows the *dimensionality* of the data, *i.e.*, the number of data features, and that features are scaled in the range $[0, 1]$, which is a standard pre-processing step. Additionally, note that our data extraction can only be applied to Neural Networks (NNs) that contain at least one fully-connected layer.

Our adversarial initialization causes the ReLU function for many neurons at the fully-connected layer to activate only for one input data point per mini-batch. Due to the randomness in the initialization of each weight row corresponding to a neuron, different neurons are likely to be activated by different input data points. Thereby, the gradients of different weight rows allow for the extraction of different individual data points. However, there are also cases where the same data point can activate different neurons. We experimentally demonstrate the success of this method in Section 4.5 by showing that our adversarial weight initialization increase the proportion of neurons that only activate on one random individual data point in a mini-batch by more than factor 10, and thus we are able to extract more than double the number of individual training data points. *E.g.* our adversarial initialization causes 51.4% of active neurons out of 1000 to by activated by individual data points from the ImageNet dataset while random model weights with a Gaussian normal initialization with $\sigma = 0.5$ only yield 4.4%. This allows for an individual extraction of 45.7% of the data points in a mini-batch of size $B = 100$ for our adversarial initialization versus 21.8% with random model weights.

We expect that an attacker with more precise background knowledge on the distribution of the training data will be able to craft even more targeted weight initializations for more effective individual data extraction. Such background knowledge may include the distribution of feature values and their relation to each other for the data to be extracted.

4.4 Adversarial Weight Initialization for Data Forwarding

So far, both the passive and active attacks we described are tailored to extracting data from the gradients computed to update the first and fully-connected layer within an ML model. However, modern NN architectures often rely on convolutional layers to model image and text data alike. It is difficult to directly apply our attack strategy to these convolutional layers because they rely on the weight sharing principle: to decrease the effective number of parameters that need to be trained, the same weight values are applied to multiple locations of the image to extract patterns regardless of their location in the image. In this section, we thus propose a second type of adversarial initialization that generalizes extraction attacks to Convolutional Neural Networks (CNNs) by forwarding the ML model's input data unaltered over convolutional layers. This allows to perform extraction from subsequent fully-connected layers as described in Section 4.3.2. Furthermore, we introduce an adversarial initialization to forward input data also unaltered



(a) 2D Example. (b) 3D Example.

Figure 4.3.: **Size-Preserving Adversarially Initialized Convolutional Filters.** Adversarially initialized convolutional filters that transmit their input to the next layer. The numbers indicated in the input and feature map represent the features, the numbers in the filter represent the weight initialization. Grey layers indicate random weight values while white fields indicate zero weights [30].

over fully-connected layers. This enables to perform data extraction at different fully-connected layers than the first one. For both our adversarial initializations, we also discuss ways to make them more inconspicuous. The results presented in this section, apart from Section 4.4.1.2 and the detailed discussion on hiding the adversarial initialization in Section 4.4.2 are taken from [30].

4.4.1 Forwarding over Convolutional Layers

Our solution reduces NNs with convolutional layers to the setting we previously considered with FC-NNs. To do so, we observe that a CNN typically composes a few convolutional layers with fully-connected layers. We thus initialize the weights of the convolutional layers such that they transmit the model input *unaltered* up to the fully-connected layers of the model architecture. We then extract the input from gradients computed for the first fully-connected layer that follows convolutional layers, using the attack strategy described in Section 4.3.

There are two important requirements for our approach to transmitting, or forwarding, model inputs through convolutional layers. The first is to make sure that no feature of the input data is lost. This requires having at least as many parameters at every convolutional layer as the number of input features. The second is to make sure that different features do not get overlaid. We explain how to ensure this in the following. Therefore, we consider two scenarios, first a forwarding over convolutional layers that preserve the size of the input data and second, convolutional layers that reduce size. The latter one can be used to make the intermediate outputs of our adversarially initialized CNNs match standard architectures more closely. At the same time, they can replace model layers for dimensionality reduction, such

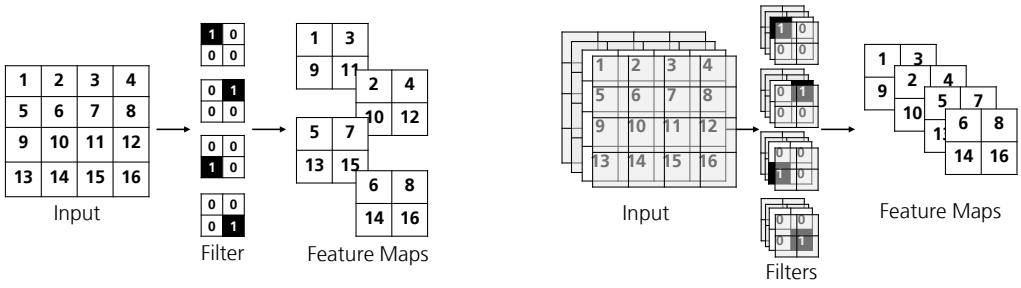
as pooling layers, which would reduce the fidelity of our extraction as we show in the following Section 4.5.

4.4.1.1 Preserving Input-Size

Two Dimensional Input. In general, preserving input size over a convolutional layer can be achieved through an adequate combination of padding, stride, and filter sizes. Specifically, we use stride one and an adequate zero-padding to preserve the size of the layer input. In order to transmit the input features, we create a filter with uneven dimensions (w, h), where $w = h$, and we initialize it with *zero* everywhere apart from the element in the middle which we set to *one*. For a two dimensional input (*e.g.* a grey-scale image), the described filter perfectly transmits the information to the next layer and creates a feature map that exactly replicates the input. See Figure 4.3a for this adversarially initialized filter.

Three Dimensional Input. Some input data to CNNs is distributed over several input channels, such as color images, that consist of three channels. At every layer, we, therefore need three adversarially initialized convolutional filters to "transmit a copy" of the input channels. A standard architecture can have many more filters per layer, which can, in the case of our attack be randomly initialized since they will be ignored by the attacker. Assume now that the original input features at the current layer l_i are distributed over a_{l_i} of the total $b_{l_{i-1}}$many feature maps. For example, in the first model layer, $a_{l_i} = b_{l_{i-1}}$ corresponds to the number of color channels required to encode the image. In subsequent layers, the remaining $b_{l_{i-1}} - a_{l_i}$ many feature maps contain random noise, introduced by random filters that do not transmit the input features (*e.g.* upper and lower filter in Figure 4.3b). We denote the indices of the feature maps where the input features are located by $\vec{\alpha_{l_i}}$. We then need a_{l_i} many filters, initialized as described above to transmit the information to the next layer. The filters differ from each other only by the placement of the matrix that contains the *one* element. This placement must correspond to different indices in $\vec{\alpha_{l_i}}$. See Figure 4.3b for a visualization of this setting. Note that the placement of the feature-transmitting filters at layer l_i will determine the indices $\vec{\alpha_{l_{i+1}}}$ of the feature maps that are input to the next layer.

In the last convolutional layer before the fully-connected layer that we want to transmit the input to, the filters containing noise should be initialized such that they yield negative input to the ReLU function. Thereby, the output of the last convolutional layer becomes zero everywhere apart from the feature maps produced by the filters transmitting input data features. The flattened output then serves as input to the fully-connected layer, where extraction can be conducted.



(a) 2D Example. (b) 3D Example.

Figure 4.4.: **Size-Reducing Adversarially Initialized Convolutional Filters.** Adversarially initialized convolutional filters that transmit their input to the next layer. The numbers indicated in the input and feature map represent the features, the numbers in the filter represent the weight initialization. Grey layers indicate random weight values while white fields indicate zero weights. Feature maps in the 3D example that only contain noise are suppressed for improved visualization.

4.4.1.2 Reducing Input-Size

Two Dimensional Input. The reduction of size in convolutional layers can be achieved by increasing the stride. However, thereby, the number of features in the next feature map is reduced such that this feature map cannot accommodate all features from the previous layer. To overcome this, we propose distributing the features of one input feature map over several feature maps in the following layer. Figure 4.4a depicts this approach for two-dimensional inputs. Note that the stride is set to the dimensions of the convolutional filters (w, h) to prevent features from overlapping in the following layer. Additionally, to transmit all the features, the dimensions of the filters w and h $(w = h)$ need to be integer dividers of the previous feature map's dimensions. Finally, in total, for each layer that reduces the size of the input by a factor $\frac{1}{w}$, we require w^2 many filters to transmit every feature from one input feature map. Hence, assuming that at layer l_i the original features are distributed over a_{l_i} many input feature maps, we require $a_{l_i} \cdot w^2$ many filters to transmit all original input features.

Three Dimensional Input. The same approach as for the two dimensional input can be extended to the case with three input dimensions. The approach is visualized in Figure 4.4b. For improved visualization, we do not present the feature maps in the layer's output which contain only noise. Again, in both the two and three dimensional case, in the last convolutional layer before flattening, the noise filters should produce negative input to the ReLU function. This enables only extracting the original input features and no noise from the following fully-connected layer.

4.4.1.3 Reducing Detectability

In principle, our adversarial weight initialization for CNNs only requires the number of filters per layer that actually transmit the features. However, using only a small number of filters, *e.g. one* as in the case of the size-preserving adversarial convolutional filters, leads to models architectures that deviate strongly from standard architectures. Therefore, we propose using a standard number of convolutional filters in every layer and initializing the filters that are not used to transmit features at random. Additionally, to prevent the simple detection strategy which relies on probing after every convolutional layer whether its input is equal to its output, one can replace the ones in the adversarially initialized convolutional filters by other positive constants. Data extraction at the fully-connected layer then yields data points where features of the original input data are scaled by (multiple different) factors. By applying the inverse of the factors encoded in the model weights this scaling can then be reverted. As a consequence, the rescaled extracted data points still perfectly correspond to the input data. Yet, the manipulations remain detectable for clients who are aware of the attack.

4.4.2 Forwarding over Fully-Connected Layers

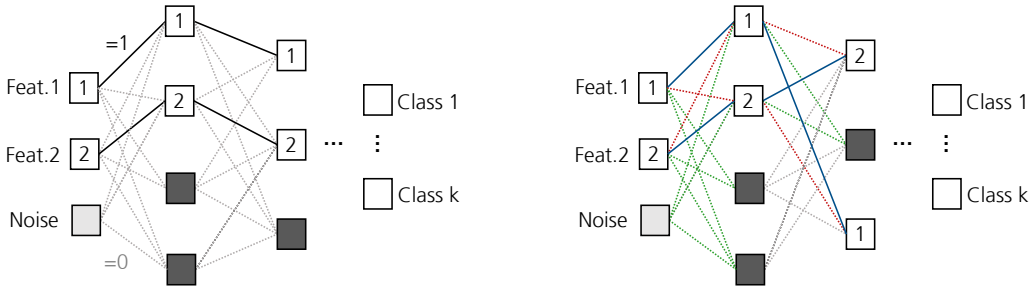
In this section, we show how to initialize a fully-connected model layer adversarially, such that it transmits its input data to the next layer without altering the it. This can be useful, for example, if a later layer in the NN has more neurons, which allows for extracting more data, as we will show in the next section. We then describe how to make such an adversarial weight initialization more inconspicuous.

Assuming that the fully-connected model layer has at least as many neurons as there are input data features, a very simple initialization would be the following: Set all components in the weight matrix at that layer to *zero*, apart from one component per input data feature, which is set to *one* in order to transmit the corresponding feature to the next layer. Thereby, model input could potentially be transmitted all the way from the model's input layer to the output layer. This setup is visualized in Figure 4.5a. In the resulting model, all neurons that do not carry an input feature would have a zero in- and output, and, thereby act as dead neurons.

4.4.2.1 Reducing Detectability

A layer that is initialized like the described prototype of our adversarial initialization would be detectable by a client aware of the method. It is, however, possible to hide the adversarial initialization and make it more indistinguishable from a random model initialization. Therefore, we implement the following ideas:

1. The first feature does not need to always be transmitted to the first neuron in the subsequent layer, instead, each feature can be transmitted to a random separate neuron. When extracting the original input data, an attacker only



(a) Prototype of an adversarial initialization forwarding input data over fully-connected layers.

(b) Different types of model weights for a more inconspicuous adversarial initialization.

Figure 4.5.: **Forwarding over Fully-Connected Layers.** Adversarial weight initializations that enable to forward an ML model's input data unaltered over fully-connected layers. Types of neurons: O_{info}, O_{noise} O_{dead}. Types of weights: (a) one and zero (b) pos, zero, neg, and random.

requires knowledge on the indices where each input data feature is located at the current fully-connected layer.

2. The weights that transmit features do not need to be set to *one*, but can be set to any random positive value. Thereby, at the layer where the input should be transmitted to, this input arrives scaled feature-wise by a factor depending on all the positive values that the feature was multiplied with on the way. Hence, after extracting the input data from the gradients at that layer, it is only necessary to rescale it feature-wise according to the inverse of the factors.
3. The model weights do not need to be all set to *zero* and *one* elements. Instead, it is possible to reorder existing model weights that were initialized with values from, for example, a Gaussian normal distribution and just replace a few of them by *zero* elements.

In the following, we explain the last point more in detail. Therefore, we denote neurons that transmits an input feature as *Informative Neurons* O_{info}, neurons that do not carry features as *Noisy Neurons* O_{noise}, inactive neurons, *i.e.*, neurons that always output zero as *Dead Neurons* O_{dead}. Additionally, we introduce four different categories of model weights:

Feature Transmitting Weights (pos) are the weights that connect the O_{info} at layer l_i carrying feature $x_i^{(j)}$ with the respective O_{info} at layer l_{i+1}. These weights need to be set to positive values to yield a positive input into the ReLU activation function. Thereby, the a scaled version of feature $x_i^{(j)}$ is transmitted over the layer.

Feature Separating Weights (zero) are the weights that connect the O_{info} at layer l_i carrying feature $x_i^{(j)}$ with all the O_{info} carrying information about any other feature

$x_i^{(g)}$, $g \neq j$ at layer l_{i+1}. These weights need to be set to zero. Otherwise, information of several features would overlay, and the data points could not be clearly extracted anymore from the gradients.

Feature Deactivating Weights (neg) are weights that connect any O_{info} at layer l_i with any O_{noise} at layer l_{i+1}. By setting these weight to negative random values, we can convert the O_{noise} at layer l_{i+1} into O_{dead} which always output zero, independent of their input data.

Random Weights (random) are weights that connect any O_{dead} at layer l_i with any O_{info} or O_{noise} at layer l_{i+1}. Since dead neurons always output zero, the product of the weights and their output will be zero independent of the input data. Thereby, these weights cannot cause an overlay of any information. Hence, the random weights can be set to any positive or negative value from the random weight initialization.

See Figure 4.5b for a visualization of how the four types of model weights can be applied to yield a more inconspicuous adversarial weight initialization for forwarding over fully-connected layers.

We now analyze how many weights of each of the four types we need. To transmit m features, we need at least **#pos** $= m$ positive weight components. To prevent features from overlaying, we need **#zero** $= m \cdot (m-1)$ zero weight components. The number of negative weight components needed at layer l_i depends on the number of O_{noise} at layer l_{i+1} which we denote here by o. At most, we need **#neg** $= m \cdot o$. Note that, in practice, it is sufficient to only set the majority of the Feature Deactivating Weights to negative components. The remaining can be set to random values as long as the weighted input to the ReLU function of the O_{noise} neurons at layer l_{i+1} is still negative, and thereby yields dead neurons. The number of random weights at layer l_i with a total of o_{l_i} neurons is then determined by **#random** $= o_{l_i} -$ **#pos** $-$ **#zero** $-$ **#neg**.

When we assign the positive, negative, and random weight components in our adversarial weight initialization with values from a standard random weight initialization, our manipulation becomes less detectable. The only elements that are still detectable are the zero weights which do not naturally occur in a standard random initialization. Additionally, if there are many O_{noise} at a given layer, we might need a large number of negative weights. This can increase detectability of our adversarial weight initialization since standard random weight initializations are zero-centered and hence contain roughly the same number of positive and negative components. However, if the attacker can control the model architecture, they can prevent such disbalance by choosing the number of neurons per model layer adequately.

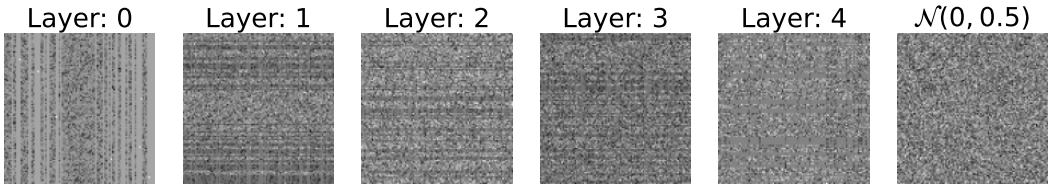


Figure 4.6.: **Adversarially Initialized Model Weights.** 100x100 pixel cutouts of the weight matrices of an FC-NN initialized with our adversarial weight initialization in comparison to a standard Gaussian normal initialization. The FC-NN consists of six layers with 1000, 3000, 3000, 2000, 1000, and 10 neurons, respectively. The adversarially initialized weights of the first five layers are depicted. The input data holds 784 features.

Figure 4.6 visualizes the weight matrices of an FC-NN initialized with our adversarial weight initialization for forwarding in comparison to a standard Gaussian normal initialization. The considered FC-NN consists of six layers with 1000, 3000, 3000, 2000, 1000, and 10 neurons. We assume 784 input features which corresponds to data from, for example, the MNIST dataset. The model is adversarially initialized to forward its input data to its penultimate layer. Hence, the last layer can be initialized completely at random, and therefore, we do not depict it in the figure. The first model layer uses only **zero**, **pos**, and **neg** but no **random** elements to transmit features and deactivate all O_{noise} in the next layer. Due to the proportionally large number of zero weight components, the changes at this first layer are most perceptible. In the following layers they are less detectable since we can assign more random weight components due to the O_{dead} neurons from their previous layers. Within all layers, the figure shows some columns containing constant values. These are the *Feature Separating Weights*, *i.e.*, the **zero** elements that are required in order to prevent that features spread over more than one O_{info} and therefore overlay.

l

4.5 Experimental Evaluation

In this section, we present the experimental evaluation of our adversarial weight initializations. Therefore, we use three different image datasets, namely MNIST, CIFAR-10, and ImageNet and the text-based Internet Movie Database (IMDb) dataset for sentiment analysis. For a thorough description of these datasets, see Section 2.4. The results presented in this section are taken from [30].

Because our approach is applicable to FC-NNs and CNNs, we test it against both of these architectures. We instantiate our attack against an FC-NN for the MNIST dataset, and against a CNN for CIFAR-10 and ImageNet. For the IMDb dataset, we use a model whose input is 250-token sentences, and consists of an *embedding*

FC-NN Architecture	VGG-inspired CNN Architecture
Dense(n=1000, act=relu)	Conv(f=128, k=(3,3), s=1, p=same, act=relu)
Dense(n=3000, act=relu)	Conv(f=256, k=(3,3), s=1, p=same, act=relu)
Dense(n=3000, act=relu)	Conv(f=512, k=(3,3), s=1, p=same, act=relu)
Dense(n=2000, act=relu)	Flatten
Dense(n=1000, act=relu)	Dense(n=1000, act=relu)
Dense(n=#classes, act=None)	Dense(n=#classes, act=None)

Table 4.1.: **Model Architectures for Experimental Evaluation of Vision Datasets.** Architectures of models used in the experiments on image data. f: number of filters, k: kernel size, s: stride, p: padding act: activation function, n: number of neurons [30].

layer, which maps each token in a 10,000-word vocabulary to a 250-dimensional floating-point vectors, and inputs these to a fully-connected layer. The specifics of our model architectures for image and text data are described in Table 4.1 and Table 4.2, respectively. We implemented our adversarial initializations and the experiments in TensorFlow [1] version 2.4.

Attack Instantiation. Because the server has access to the gradients of *all* model layers uploaded by users, it is able to choose which layer to instantiate the attack on. For the FC-NN, we adversarially initialize the first layer and extract training data points from its gradients. Through our initialization that allows for transmission of input data over fully-connected layers (see Section 4.4.2), we could also forward the input data to a later layer and perform extraction there. As stated in the previous section, forwarding to a later fully-connected layer might be useful when this layer consists of more neurons than the first fully-connected layer: The experimental results in Table 4.7 show that the more neurons, the more data points can be extracted individually.

In the CNNs, we first initialize the convolutional layers to transmit the input data to the first fully-connected layer of the architecture. Then, we adversarially initialize this layer's weights for extraction.

For the text classifier, we initialize the weights of the embedding layer with a random uniform distribution (min=0., max=1.) to create the inputs for the fully-connected layer. We then adversarially initialize this fully-connected-layer's weights to perform extraction of the embeddings there. Finally, after extracting the input's embeddings from then fully-connected layer (like they would extract an image), we map them back to the corresponding text. To reconstruct the original text tokens from a sequence of extracted embeddings, the attacker creates a lookup dictionary, mapping its initialized embeddings back to their corresponding tokens (this is the

IMDb-Model Architecture
Embedding(feat=10000, dim=250)
Dense(n=1000, act=relu))
Dense(n=1, act=None)

Table 4.2.: **Model Architecture for Experimental Evaluation of Text Dataset.** Architecture of models used in the experiments on the IMDb dataset. feat: vocabulary size, dim: embedding size, act: activation function, n: number of neurons [30].

inverse mapping to the embedding layer). To avoid vector-comparisons for each lookup, we use hash values for vector embeddings as keys.

4.5.1 Extraction Success Metrics

We introduce three novel metrics to measure the success of individual data point extraction from model gradients.

Active Neurons. By measuring the number of *active neurons* (A) we can determine for how many neurons their respective weighted inputs are positive. This is important because data extraction for both overlaying and individual data points is only possible with activated neurons. If a neuron is not activated by any data point, no information can be transmitted over this neuron and, hence, the gradients will all be zero.

Extraction-Precision. Our second metric, which we call *extraction-precision*, captures the percentage of non-zero gradient rows at the given layer's weight matrix from which we can extract any input data point *individually*. This metric enables us to quantify how well the adversarial weight initialization manages to generate weights that cause activation for exactly one single data point. *Extraction-Precision* can be calculated as follows:

$$\mathrm{P} = \frac{G_1}{A}, \tag{4.8}$$

with A denoting the *active neurons*, and G_1 denoting the number of gradient rows from which we can extract a data point individually and with an ℓ_2-distance of zero to any of the input data points.

However, the *extraction-precision* metric alone would not be expressive enough since a high *extraction-precision* could be achieved despite the exact same individual training input being reconstructed from all gradient rows. Therefore, we defined another metric that we call *extraction-recall*.

	MNIST			CIFAR-10			ImageNet		
Weights Initializer	A	P	R	A	P	R	A	P	R
Xavier Normal	.998	.004	.037	.810	.039	.190	.942	.046	.213
Xavier Uniform	.998	.005	.048	.808	.040	.192	.945	.040	.201
Gaussian (σ=0.01)	.997	.005	.048	.807	.042	.193	.940	.041	.203
Gaussian (σ=0.1)	.997	.005	.049	.807	.042	.195	.940	.043	.209
Gaussian (σ=0.5)	.997	.006	.050	.807	.046	.203	.940	.044	**.218**
Gaussian (σ=1)	.997	.006	.059	.807	.047	**.206**	.940	.045	**.218**
Gaussian (σ=2)	.997	.007	**.061**	.806	.047	**.206**	.000	.000	.000

Table 4.3.: **Extractability with Random Initializations.** Impact of random initialization functions on the *extraction-precision* (P) and *extraction-recall* (R) of individual training data points from the model gradients. The displayed numbers refer to a mini-batch of 100 data points and 1000 neurons for extraction in the first model layer (FC-NN architecture from Table 4.2). Results are averaged over 10 runs with different random initializations [30].

Extraction-Recall. The *extraction-recall* measures the percentage of input data points that can be perfectly extracted from any gradient row. We define it by

$$\mathrm{R} = \frac{B_0}{B}, \tag{4.9}$$

where B is the number of data points in the given mini-batch and B_0 is the number of these data points that we can extract with an ℓ_2-error of zero from the rescaled gradients.

Interpretation of Success Metrics. Note that our attack seeks to find an adversarial initialization that balances setting enough neurons' outputs to zero (such that a gradient is more likely to isolate individual points from large mini-batches) with, at the same time, having enough neuron outputs' that are non-zero (otherwise, in the limit, no points would be extracted). Thus, *active neurons* provide additional context for the *extraction-precision*: with few *active neurons*, even a high *extraction-precision* might not be able to extract many individual training data points, simply because there are very few gradients to perform data extraction from. However, with many *active neurons*, the *extraction-recall* might become small, due to each neuron being most likely activated by several input data points, preventing individual extraction.

4.5.2 Evaluating Passive Data Extraction

Recall from Section 4.2 that extraction of training data from gradients is possible even when model weights are initialized randomly. We evaluate this passive extractability to obtain a baseline for our adversarial weight initialization strategies. To evaluate the passive extractability, *i.e.*, the passive attack, we measure the ex-

	Passive Attack			Our Active Attack		
B	A	P	R	A	P	R
20	.842	.072	**.900**	.519	.610	**1.000**
50	.885	.050	**.552**	.776	.376	**.962**
100	.909	.036	**.254**	.910	.192	**.654**
200	.927	.030	**.128**	.978	.070	**.255**

Table 4.4.: **Data Extraction on IMDb Dataset.** The extraction success depends on the size **B** of the mini-batches for passive attack and active attack with adversarial initialization. The results depict the percentage of *active neurons* (A), *extraction-precision* (P), and *extraction-recall* (R). All numbers are averaged over 10 runs with different random and adversarial initialization of the model from Table 4.2, respectively [30].

traction success of individual training data points from the gradients of randomly initialized models.

Table 4.3 reports the results of training data point extraction from the gradients of randomly initialized models. These gradients are computed over a mini-batch of 100 data points for 1000 neurons (*i.e.*, 1000 weight rows' gradients for extraction) in the first fully-connected layer. We later study the impact of these two parameters (mini-batch size and number of neurons) on the success of reconstruction attacks. Even if this data extraction can be considered as a passive attack without any model weight manipulations, training data extraction is often successful: for the MNIST dataset, around 6% of individual training data points can be directly extracted from the model gradients, whereas for CIFAR-10 and ImageNet, roughly 21% and 22% of the training data points can be perfectly extracted, respectively. The passive attack for extracting embeddings from the IMDb dataset yields roughly 25% *extraction-recall* for 1000 neurons and mini-batches of 100 data points, see Table 4.4.

The results from Table 4.3 also suggest that using a normal distribution, instead of a uniform one, and setting higher standard deviations in random weight distributions alone can already significantly increase the *extraction-recall* of individual data points from the model gradients. This is, most likely, due to the larger span within the weight values.

Additionally, we also set out to investigate how as training progresses, and the model's weights converge, the extraction' success evolves. We initialized the FC-NN from Table 4.1 with a Xavier Uniform distribution and trained the model on MNIST and CIFAR-10 for 30 epochs. Table 4.5 depicts the results. We observe that the *extraction-recall* increases slightly over the training epochs. Analyzing the distribution of the model weights in Figure A.1 in Appendix A.3 shows that over training, the uniformly initialized weight values resemble more a normal

	MNIST				CIFAR-10			
Epoch	Loss $\mathcal{L}$	A	P	R	Loss $\mathcal{L}$	A	P	R
0	.526	.998	.005	.050	1.857	.907	.053	.232
5	.067	.997	.044	.137	1.352	.900	.044	.195
10	.021	.997	.116	.154	1.088	.913	.041	.196
15	.006	.997	.131	.165	.768	.923	.043	.206
20	.002	.997	.136	.167	.472	.931	.050	.232
25	.001	.997	.140	**.169**	.282	.935	.058	.241
30	.001	.997	.142	.168	.200	.936	.062	**.267**

Table 4.5.: **Data Extractability from Converging Models.** Results depict the success of passive data extraction based on the training stage of the corresponding models. We show the percentage of *active neurons* (A), *extraction-precision* (P), and *extraction-recall* (R) for extraction with a mini-batch size of 100 data points from the first layer of the fully-connected network from Table 4.1. All numbers are averaged over 10 runs with different *random* initializations [30].

distribution and obtain a wider spread, which might be the reason for the increased extraction success.

4.5.3 Evaluating Active Adversarial Weight Initialization

We now turn to our active attack, which adversarially modifies the weight initialization to amplify the vulnerability exploited by passive attacks. This amplification is controlled by the scaling factor s. We first evaluate its impact on the reconstruction quality of individual training data points over a mini-batch of 100 data points and 1000 neurons.

For all our results, we report the *active neurons*, *extraction-precision*, and *extraction-recall*. Table 4.6 depicts the results, averaged over ten different adversarial initializations. We can see that the best scaling factor for MNIST, when it comes to the *extraction-recall*, is $s = 0.7$. With this scaling factor, we are able to extract on average 54.0% of the individual training data points which were involved in the clients' gradient computations. This is an improvement by around factor nine to the passive attack. For CIFAR-10 and ImageNet, the best scaling factors concerning *extraction-recall* are $s = 0.95$, and $s = 0.99$, which allow for a perfect reconstruction of 54.0%, and 45.7% of the individual training data points, respectively, for 1000 neurons and a mini-batch size of 100 data points. Thereby, the active attack is more than twice as successful as the passive attack for extracting individual training data points in these datasets. Figure A.2, Figure A.3, and Figure A.4 in the Appendix A.3 show the visual reconstruction results of the best runs for all three image datasets and the respective hyperparameters. Similar improvements of performance could be achieved for the IMDb dataset. The best extraction was achieved also with $s = 0.99$,

	MNIST			CIFAR-10			ImageNet		
s	A	P	R	A	P	R	A	P	R
.400	.022	.803	.114	0.	0.	0.	.0.	0.	0.
.500	.149	.636	.354	0.	0.	0.	0.	0.	0.
.600	.462	.408	.526	0.	0.	0.	0.	0.	0.
.700	.796	.203	**.540**	0.	0.	0.	0.	0.	0.
.800	.959	.062	.334	0.	0.	0.	0.	0.	0.
.900	.996	.010	.089	.034	.946	.077	0.	0.	0.
.950	.999	.003	.029	.729	.412	**.540**	0.	0.	0.
.960	.999	.003	.027	.925	.175	.522	0.	0.	0.
.970	1.	.002	.020	.993	.025	.198	.002	.900	.013
.980	1.	.002	.021	1.	.001	.008	.043	.986	.049
.990	1.	.002	.020	1.	0.	0.	.655	.514	**.457**
.995	1.	.002	.018	1.	0.	0.	.999	.007	.055
.999	1.	.002	.017	1.	0.	0.	1.	0.	0.

Table 4.6.: **Impact of Hyperparameter s.** Success of our adversarial weight initialization dependent on the hyperparameter s, which downscales the positive weights. The results depict the percentage of *active neurons* (A), *extraction-precision* (P), and *extraction-recall* (R) with a mini-batch size of 100 data points from the first fully-connected layer of the respective architectures from Table 4.1. All numbers are averaged over 10 runs with different adversarial initializations [30].

for which, with 1000 neurons and mini-batches of 100 data points, we obtained an *extraction-recall* of 65.4%, which is around 2.5 time as high as the passive attack, see Table 4.4 on page 67.

As hypothesized above, we furthermore confirm that the *extraction-recall* of our attack is related to the percentage of *active neurons*: When very few neurons are activated, it is not possible to extract large numbers of individual data points due to the lack of gradients to extract them from. However, when the percentage of *active neurons* is high, the *extraction-recall* also becomes very small, which is due to the fact that each neuron gets activated by several input data points, and thereby, individual extraction is impossible.

Impact of Data Labels. Additionally, we investigated whether this high reconstruction success could also be achieved when all data points in the mini-batch belong to the same class. This is a particularly challenging setting for prior work on optimization-based attacks that end up reconstructing average points rather than individual points exactly. Instead, Figure A.3b in the Appendix A.3 shows how, on CIFAR-10, our method remains able to perfectly extract individual data points from the gradients even when all points stem from the same class.

	MNIST			CIFAR-10			ImageNet		
(B, N)	A	P	R	A	P	R	A	P	R
(200, 20)	.522	.436	**.720**	.454	.670	**.695**	.090	.948	**.355**
(200,50)	.690	.302	.428	.662	.494	.452	.381	.763	.304
(200,100)	.782	.196	.218	.846	.280	.269	.653	.500	.240
(200,200)	.859	.121	.086	.954	.124	.096	.886	.233	.113
(500,20)	.535	.451	**.915**	.452	.689	**.870**	.096	.939	**.490**
(500,50)	.697	.301	.624	.653	.505	.614	.387	.767	.426
(500,100)	.792	.205	.397	.845	.290	.422	.646	.508	.358
(500,200)	.871	.129	.185	.950	.119	.177	.892	.240	.199
(1000,20)	.539	.444	**.950**	.441	.703	**.915**	.102	.942	**.595**
(1000,50)	.705	.300	.760	.648	.504	.724	.388	.770	.516
(1000,100)	.796	.203	.540	.844	.297	.556	.655	.514	.457
(1000,200)	.871	.124	.293	.951	.120	.256	.892	.238	.288
(3000,20)	.541	.442	**1.**	.441	.696	**.945**	.101	.934	**.640**
(3000,50)	.704	.299	.888	.646	.503	.812	.386	.764	.586
(3000,100)	.797	.203	.746	.840	.286	.711	.649	.518	.579
(3000,200)	.873	.129	.504	.951	.122	.414	.889	.243	.404

Table 4.7.: **Effect of Mini-Batch Size and Number of Neurons on Data Extraction.** Success of our adversarial weight initialization dependent on the mini-batch size **B** and the number of neurons **N** that corresponds to the number of weights rows. The results depict the percentage of *active neurons* (A), *extraction-precision* (P), and *extraction-recall* (R). All numbers are averaged over 10 runs with different adversarial initializations.

Impact of Mini-Batch Sizes. We also set out to investigate the impact of the mini-batch size B and the number of weight rows that we can use for extraction. Table 4.7 depicts the resulting metrics. The metrics show that the smaller the mini-batch sizes are, and the more weight rows there are for extraction, the more individual training data points can be individually reconstructed. For 3000 weight rows, even up to 50% of the individual training data points for mini-batch sizes as large as 200 in the MNIST dataset can be perfectly extracted. Small mini-batches of 20 training data points are entirely extractable without any loss in this setting. Also for the IMDb dataset, smaller batch-sizes for the same number of neurons yield much higher *extraction-recall*, and embeddings of data from small mini-batches of 20 training data points are perfectly extractable, see Table 4.4. This suggests that in practice, the success of the extraction attack can be significantly increased by the server demanding smaller mini-batch sizes from the clients or initializing larger models.

Impact of Mini-Batch-Averaging. Additionally, we looked into the effect of local averaging over the gradients of multiple mini-batches before extraction. The results in Table 4.8 show that through averaging, the attack success is significantly reduced. Already when averaging over 20 mini-batches of size $B = 100$ in the MNIST dataset,

B, Num	A	P	R
(20,1)	.496	.486	.950
(20,5)	.787	.213	.572
(20,10)	.851	.157	.412
(20,20)	.898	.116	.251
(50,1)	.687	.307	.790
(50,5)	.901	.107	.230
(50,10)	.928	.080	.138
(50,20)	.953	.053	.067
(100,1)	.800	.200	.562
(100,5)	.936	.066	.116
(100,10)	.966	.046	.054
(100,20)	.982	.028	.020

Table 4.8.: **Effect of Mini-Batch Averaging.** Success of our adversarial weight initialization on MNIST under averaging over multiple mini-batches on the same model parameters. The number of mini-batches is denoted by **Num** and their respective size by **B**. The results depict the percentage of *active neurons* (A), *extraction-precision* (P), and *extraction-recall* (R) for extracting from 1000 neurons at the first layer of the FC-NN depicted in Table 4.1. All numbers are averaged over 10 runs with different adversarial initializations [30].

the average *extraction-recall* drops from 54.0% to 2.6% because multiple data points overlay in the gradients.

Impact of Lossy Layers. We also studied the effect of "lossy" layers, such as dropout and pooling on our data extraction success. Therefore, we relied on a dedicated architecture proposed by [5] for FL, see Table 4.9. We evaluated the effect of dropout

CNN Architecture by [5]
Conv(f=32, k=(3,3), s=1, p=same, act=relu)
MaxPool()
Conv(f=64, k=(3,3), s=1, p=same, act=relu)
Dropout()
Flatten
Dense(n=1000, act=relu)
Dropout()
Dense(n=#classes, act=None)

Table 4.9.: **CNN Architecture with Lossy Layers.** CNN Architecture by [5] used to evaluate the data extraction attack under the impact of dropout and pooling. f: number of filters, k: kernel size, s: stride, p: padding act: activation function, n: number of neurons.

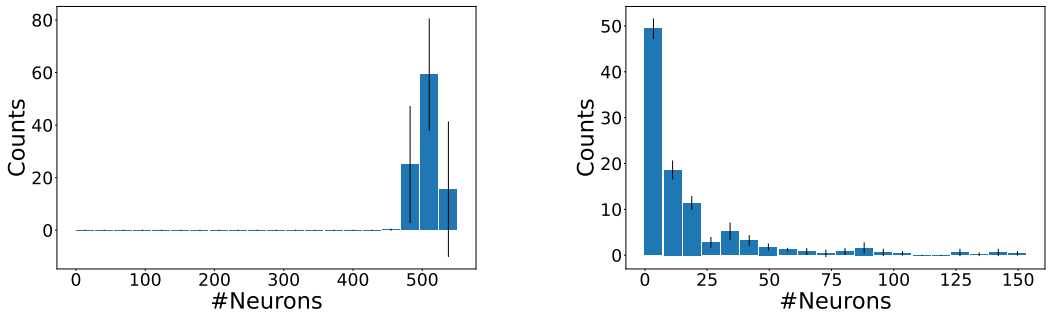
(a) Randomly Initialized Weights.

(b) Adversarially Initialized Weights.

Figure 4.7.: **Individually Extracted Data Points and Occurrences from CIFAR-10.** Number of individual occurrences in the rescaled gradients over a mini-batch of 100 data points, extracted at the first fully-connected layer of the CNN from Table 4.1 in Section 4.5. To provide insights into what data points could not be individually extracted, we plot data points with zero occurrences with low saturation.

and pooling layers for mini-batch sizes of one and 20. The dropout rates were chosen from $p \in \{0.0, 0.1, 0.3, 0.5, 0.7, 0.9\}$. Figures A.5 and A.6 and Figures A.7 and A.8 in Appendix A.3 show individual effects of dropout and pooling layers on a reconstructions for mini-batches of size 1 and 20, respectively. Note that the second dropout layer does not have a significant impact on the success of our reconstruction since we extract from the first fully-connected model layer before information can get lost due to the second dropout. To evaluate dropout without pooling, we removed the MaxPool layer, and to evaluate pooling without dropout, we set the dropout rate to $p = 0.0$. The reconstructed images show that although the existence of non-invertible "lossy" components compromises overall reconstruction fidelity, it is often still possible to recognize individual data points.

Insights into Individual Extractability and Adversarial Initialization. To conclude the experimental evaluation of our adversarial weight initialization and to give an outlook on potential future improvements of our method, we present additional insights into the individual extractability of training data points. Therefore, we first studied *how often* each data point is individually extractable from the rescaled



(a) Randomly Initialized Weights.

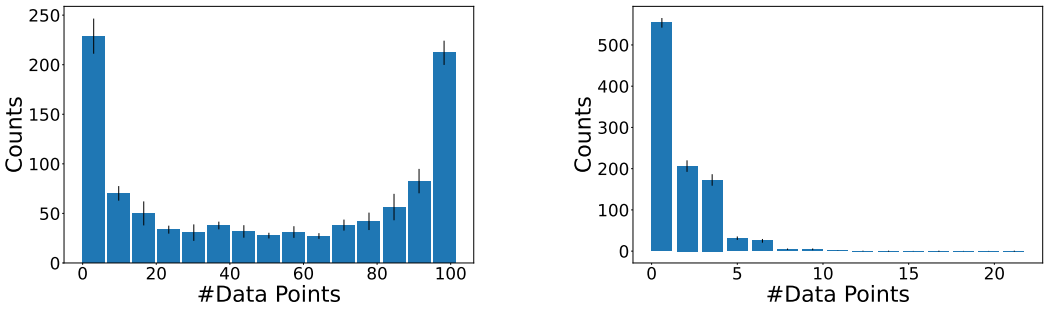
(b) Adversarially Initialized Weights.

Figure 4.8.: **Number of Neurons Activated by CIFAR-10 Data Points.** The results depict the number of neurons that each of the 100 data points in a mini-batch activates. Results are averaged over five different random and adversarial model initializations.

gradients. Figure 4.7 shows data points from a training mini-batch with 100 data points from the CIFAR-10 dataset and their respective individual occurrences in the rescaled gradients. Results are depicted for both randomly initialized model weights and our adversarial weight initialization. The results suggest that our adversarial weight initialization, first of all, make more data points individually extractable, but also cause the same data points to be individually extractable from many more different weight rows' gradients (up to 33 times over the 1000 neurons and their respective weight rows). Also note that except from very few exceptions, all data points that are individually extractable from randomly initialized weights are also extractable with our adversarial weight initialization. This highlights that our method builds up on natural leakage from gradients, increases it, and extends it to a broader range of data points.

Furthermore, to showcase the effectiveness of our adversarial weight initialization, we evaluate *how many neurons* each data point activates (*i.e.*, for how many neurons the weighted input of this data point is positive). Remember that we want to reduce the number of neurons that each input data point naturally activates to prevent overlaying data points in the rescaled gradients. Figure 4.8 highlights that while for random weights, most of the data points activate around 500 of the 1000 neurons in the fully-connected layer, with our adversarial weight initialization, most data points activate less than 25 neurons, confirming the effectiveness of our method.

Finally, we studied by how many data points each neuron gets activated. The results in Figure 4.9 suggest that our adversarial weight initialization is highly effective in causing individual activation of neurons, *i.e.*, making neurons activated by only one input data point. Potential room for improvement lies in adapting the initialization of neurons that are not activated by any data point. These neurons' potential for individual data point extractability is currently wasted. What an



(a) Randomly Initialized Weights. (b) Adversarially Initialized Weights.

Figure 4.9.: **Number of CIFAR-10 Data Points that Activate Each Neuron.** Number of data points that activate each one of the 1000 neurons. Results are averaged over five different random and adversarial model initializations.

attacker in possession of a small number of data points from the data distribution could, for example, do to overcome this limitation is to initialize the model with adversarial weights, evaluate activation of the neurons on this data, and then replace the weight rows that do not cause activation of their respective neurons with different adversarially initialized weight rows that do lead to activation on any input data point.

4.6 Conclusion

In this chapter, we presented *adversarial weight initialization* as a novel attack vector against privacy in ML. Therefore, we first formalized our original observation that in NNs starting with a fully-connected layer, even gradients of very large training data mini-batches contain *individual* training data points. Based on this observation, we showed how to use adversarial weight initializations to amplify the leakage of individual training data points and extend it to other model architectures. To experimentally evaluate the success of our novel attack vector, we performed data reconstruction on image and text data with both FC-NN and CNN models. Our evaluation highlights that our attack is able to extract large fractions of training data points perfectly, *i.e.*, with zero-error from the model gradients. In contrast to previous data reconstruction attacks that rely on computationally expensive operations, our attack enables the extraction of individual training data points in a single-step computation and is, thereby, highly efficient.

We also showed that our novel attack vector integrates naturally with model training performed through the protocol of FL (see Figure 4.1 on Page 48). Thereby, it allows the server who is in charge of the shared model and receives the client-gradients to disclose the clients' privacy and directly extract their sensitive training data. Since FL was, for a long time, promoted as a privacy-preserving solution for ML, our discovery of this novel attack vector is highly relevant and valuable. It does not

only contribute to improving our understanding of privacy leakage in ML, but it also helps to raise awareness of existing privacy threats in this widely applied framework. Furthermore, it can inform the study on trust assumptions around privacy, and the development of adequate protection measures as we will show in the following chapter.

Chapter 5

Studying Trust Assumptions in Federated Learning

Third-party services and their providers are of high relevance when it comes to ensuring the privacy of Machine Learning (ML) models as we concluded from the results of our survey in Chapter 3. This is because ML practitioners trust the service providers to implement adequate privacy protection for their sensitive data and learning tasks. Yet, the previous Chapter 4 highlighted that Federated Learning (FL), which was, for a long time promoted by its provider as a privacy-preserving ML protocol, leaks private client-data. We showed that this data leakage can even be increased when the server acts maliciously and manipulates the shared model weights. These findings motivate us to perform an in-depth study of trust assumptions underlying the FL protocol in the scope of this chapter. In particular, we aim at investigating under which trust assumptions FL can meet the promised privacy protection in practice.

In this work, we focus on trust put in the server. Building on our results from the previous chapter, we start by assessing privacy guarantees of the standard vanilla FL protocol—*i.e.*, an FL protocol that does not implement additional measures to protect its clients' privacy—in presence of an untrusted server. In this scope, we discuss both a passive *honest-but-curious* server who simply analyses the received client-gradients, and an actively *malicious* server who exhausts its abilities within the protocol to deliberately target the clients' privacy. We then move on to study the requirement for trust in the server within FL protocols that implement dedicated privacy-protection. In particular, we consider FL protocols that implement Secure Aggregation (SA) [33] and Distributed Differential Privacy (DDP) [181], currently considered the most private instantiation of the protocol. We do not consider a specific protocol variant, nor a specific proprietary system (because the details of these are usually unknown). Instead, we view the landscape of known solutions and ask what guarantees *can* be obtained.

Based on our assessment of different trust assumptions, we then set out to discuss the requirements for building a variant of FL that practically prevents attacks by a malicious server. We find that one promising direction for future research is to add adequate amounts of noise to clients' gradients via a cryptographic protocol such as Secure Multiparty Computation (SMPC). However as of yet, due to the gradients' high dimensionality, the communication costs are prohibitive. We thus conclude that *privacy-preserving FL, in the presence of an untrusted server, is not yet practical.* However, our assessment enables us to formulate some practical guidelines for reducing the trust required in the server by making data extraction attacks more difficult. These guidelines can serve clients who participate in an FL protocol and ML practitioners who are in charge of implementing it as behavioral guidance towards improved privacy for sensitive data in ML.

In this chapter, we first provide a description of the real-world FL deployments and the types of attackers that we base our study of trust assumptions on. Afterwards, we study privacy, first in vanilla FL, and then in hardened FL with DDP and SA. Based on the insights gained from assessing trust in both scenarios, we discuss different options to improve FL privacy in the presence of untrusted servers. Finally, we conclude with a summary of our findings and their implications on privacy in current FL deployments.

5.1 System Design and Threat Model

We begin this section with an overview of the real-world FL systems that underlie our assessment in this chapter, both, a standard vanilla deployment and a hardened deployment that explicitly integrates privacy-preservation. Moreover, we characterize the two broad types of attackers that we consider to study the trust assumptions in FL.

5.1.1 Real-World Federated Learning Deployment

This section describes the real-world FL deployments on which we base our evaluation of trust assumptions. To revisit the theoretical background of FL, see Section 2.2.

Standard Vanilla FL. We perform our assessment of trust assumptions based on practically deployed FL systems, such as the one described in [32]. These systems rely on synchronous large mini-batch training on the client side. For each training round of the FL protocol, a fraction of all participating clients announces their availability to the server during a certain time window. Then, the server selects a subset of typically a few hundred clients for participation. Since *"ensuring [that] devices are subsampled precisely and uniformly at random from a large population would be complex and hard to verify."* [121], we can assume that a malicious server can freely select the clients, in particular, some target client to attack in combination with

maliciously controlled clients. We assume encrypted network traffic to prevent eavesdropping and we follow [32] concerning anonymity: *"We want devices to participate in FL anonymously, which excludes the possibility of authenticating them via a user identity"*.

Hardened FL. To study hardened deployments of FL, *i.e.*, instantiations of the protocol that implement dedicated privacy-protection, we focus on FL that is extended with a combination of DDP and SA, see Section 2.3.5.3. We do so because it is the combination of published techniques that holds the most promise in the presence of an untrusted central party. Thus, our work *conservatively* characterizes the risk of privacy leakage from many instantiations of FL. Thereby, our goal in studying this particular instantiation of FL is to show that, even if servers (*e.g.*, companies) adopted currently available best practices from the academic community, end-users might not get the promised privacy protection from FL.

Keeping these considerations in mind, please also note that FL with DDP and SA is not as widely deployed as vanilla FL. This is mainly due to increased computation and communication costs, and the resulting decreased number of clients who can participate per round.[1]

5.1.2 Types of Attackers

Additionally, we introduce the different types of attackers considered in this chapter. In all scenarios, our attacker is the server in FL, and their goal is to infer individual clients' local sensitive training data points.

Honest-but-Curious. Honest-but-curious servers are *passive* attackers. They can observe the natural course of FL training but not corrupt its inputs. Thereby, their power when it comes to attacking clients' privacy in FL is limited. However, in the following section, we will show under which assumptions even an honest-but-curious attacker can extract significant fractions of the clients' training data in vanilla FL.

Malicious. Malicious servers are attackers who *actively* try to disclose the clients' privacy. Note that the background on FL in Section 2.2 implies that the server can—whenever they choose to—(1) control the weights of the shared model, (2) select which of the N clients participate in each round, and (3) provision new clients into the pool (including *sybils* controlled by the server). We have shown the attack vector that results from (1) in the previous chapter. Capabilities (2) and (3) have even been demonstrated in the real-world as Google researchers introduced 189 sybils devices into the Gboard FL system and made them participate in the protocol

[1] *"Several costs for SA grow quadratically with the number of users, most notably the computational cost for the server. In practice, this limits the maximum size of a SA to hundreds of users."* [32]. With reduced numbers of clients, also the convergence speed of the joint model training with FL decreases, *e.g.* [201].

along with real clients [152]. Given the large-scale deployment of FL protocols, one can, however, assume that the total fraction of sybils out of the N clients is small.

We, furthermore, assume the server is Occasionally Malicious (OM), meaning they behave maliciously in only a few rounds of the protocol. When this happens, they can exert the above capabilities (1-3) maliciously. Do note that the server here does not deviate from the protocol and restricts itself to only using valid operations (1-3) in the FL protocol adversarially. An OM server ensures the attack remains stealthy and also allows the server to train a model that has high utility over the non-malicious rounds, which is an expected product of FL. This is because due to its OM nature, the server can simply go back to the normal and benign training procedure after one or a few attack round(s).

5.2 Assessing Vanilla Federated Learning

We start by assessing the trust assumptions required to provide practical privacy guarantees in vanilla FL. Therefore, we begin with considering a passive honest-but-curious server, and then continue to evaluate an actively malicious one.

5.2.1 Honest-but-Curious Servers

First of all, note that nothing in the design of vanilla FL guards against leakage of private information from the clients' data. This is because the protocol is designed to provide *confidentiality* (data does not leave user devices) rather than *privacy* (outputs of the computation do not leak sensitive attributes from the clients' input). Without additional privacy measures, FL is not designed to protect clients from the server reconstructing their data.

However, as discussed in Section 4.1.2, the fidelity of prior methods' reconstructions is comparably low, in particular for more complex data. This made recent work [190] argue that privacy for vanilla FL might not be broken because these attacks have too specific assumptions on small mini-batch sizes and on the underlying data distribution.

However, our novel observation that model gradients directly leak individual training data points, even for large mini-batches of complex training data—which we formalized in Section 4.2—shows that prior work has dramatically underestimated the vulnerability of FL to attacks on privacy. This is because the data leakage can be exploited directly by an untrusted server to conduct high-fidelity data extraction attacks. This even holds when the server only passively observes the gradients, given that the shared model starts with a fully-connected layer. We thus conclude that previous privacy assessments of FL have been overly optimistic.

5.2.2 Malicious Servers

With the introduction of our adversarial weight initializations as a novel attack vector against privacy—which we presented in Section 4.3—we showed that an untrusted malicious server even has an upper hand over how much data the model gradients will leak. By adversarially manipulating the weights of the shared model, the server can extract significantly larger fractions of the clients' training data. Additionally, with the help of data forwarding (see Section 4.4) data extraction becomes feasible with a wider range of architectures, such as CNNs. This reduces the dependence on potentially limiting assumptions, such as the presence of a fully-connected layer at the input of the model.

5.2.3 Conclusion

To conclude the assessment of trust assumptions and resulting practical privacy guarantees in vanilla FL, we observe that without trust in the server, vanilla FL cannot provide privacy against both accidental and malicious leakage.

5.3 Assessing Hardened Federated Learning

In this section, we consider hardened versions of FL that aim at implementing privacy guarantees for the clients. Since these extensions were designed to protect against passive attackers, in the following, we start our assessment directly by assuming a malicious server. We, thereby, aim at understanding how much privacy can still be leaked by an attacker, even under the most protective privacy practices.

5.3.1 Differential Privacy and Trust in Federated Learning

To counteract privacy leakage, the vanilla FL protocol can be extended to implement Differential Privacy (DP) [57] guarantees. As presented in Section 2.3.5.3, there exist several ways of integrating DP in FL.

First and foremost, *if* clients do trust the server, a relatively cost-effective mitigation against privacy leakage in FL exists: the server can add noise to the model updates and, thereby, implement Centralized Differential Privacy (CDP) [152] (see Section 2.3.5.3). Therefore, the clients solely have to clip their gradients, and then the server can add noise during the aggregation [123]. This degrades the performance of learned models but adds strong privacy guarantees. However, without trust in the server, central solutions like CDP cannot provide practical privacy guarantees. This is because the server could either extract the client data before adding the noise or do not perform the noise addition, at all.

As alternative privacy protection against untrusted servers, the clients could implement Local Differential Privacy (LDP) [182] and add noise to their local gradients

before sending them out to the server. This would prevent any untrusted server from performing data extraction. Unfortunately, LDP generally results in poor model utility due to the addition of large amounts of noise to every client's update [103], and, therefore, does not represent a practical solution for FL privacy.

To overcome the shortcomings of LDP, DDP was introduced. Since it allows clients to add less noise locally but still obtain meaningful privacy guarantees through aggregation with other clients, it promises practical privacy for FL. However, in the next section, we will describe an attack that circumvents DDP and SA and allows a malicious server to still reconstruct private information from the clients. We also forge an intuition of what factors contribute most to the leakage.

5.3.2 Attacking Differential Privacy and Secure Aggregation

For successful data reconstruction under DDP and SA, the server has to make use of the following three capabilities which it naturally holds in FL:

1. *Introducing sybil devices:* The server needs to be able to introduce a fraction of manipulated devices in the FL protocol. These devices can return arbitrary gradients, chosen by the server. In particular, they can contribute *zero gradients* to the SA. By aggregating a target client's gradients with all zero gradients even within the SA protocol, this client's gradients are still perfectly extractable.
2. *Controlling the client sampling:* To ensure that the sybil devices are sampled for SA together with a chosen target client, the server needs to control the client sampling.
3. *Manipulating the model weights:* For improved data reconstruction performance, the server can manipulate the shared model's weights, for example, relying on our adversarial weight initialization method from Section 4.3.

While the first two capabilities enable the server to circumvent SA and to leave the gradients of a target client with an insufficient amount of noise for privacy protection under DDP, the third capability increases the number of individual training data points that can be reconstructed and extends the attack to other model architecture types as described in the previous chapter.

5.3.2.1 Attack Flow

The flow of our attack against FL with DDP and SA is depicted in Figure 5.1. It aims at reconstructing the private data of one target client per malicious round in the FL training. To do so, the attack needs to (1) circumvent the SA, then (2) exploit the weak privacy guarantees of DDP from a user's perspective, and (3) finally reconstruct the target client's individual training data points.

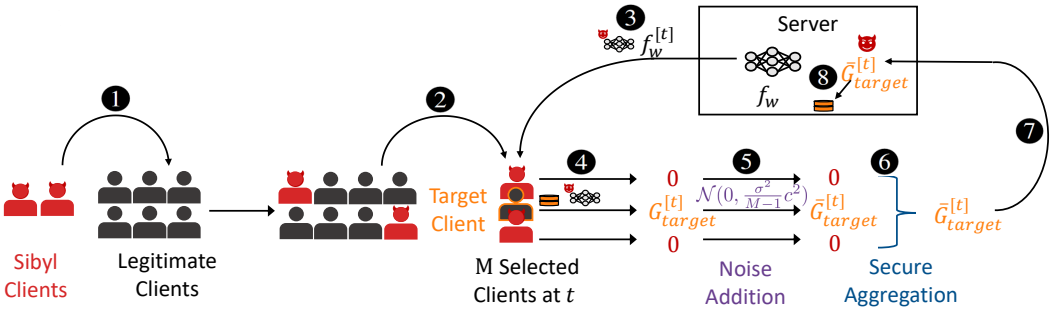


Figure 5.1.: **Attack Flow to Circumvent DDP and SA in FL.** We present the steps of our attack against FL protected by DDP and SA. ❶ The server introduces a small fraction of sybil clients into the FL application. ❷ The server selects M clients for participation in training round t: one target client and $M-1$ sybils. ❸ The server manipulates the shared model with our adversarial weight initialization and sends it out to the selected clients. ❹ The target client locally calculates its gradients on the manipulated model while the sybil clients return zero-gradients. ❺ Only the target client locally applies a small amount of noise to its gradients to implement DDP. ❻ The target client's local noised gradients are aggregated with the sybil clients' zero-gradients. ❼ The resulting aggregate which solely contains the target client's gradients is sent to the server. ❽ The server extracts the target client's training data from the received gradients.

(1) Circumventing SA. In our attack, the server circumvents SA by sampling the target client together with maliciously controlled sybil devices for the given training round. Since for each round, M clients are sampled for participation, the server needs to control at least $M-1$ sybil devices. SA limits M to several hundred clients due to the additional costs introduced by the protocol [32]. Hence, inserting $M-1$ sybil devices into an FL deployment is a practically feasible setup, as shown in previous work [152].

Since SA-protocols provide their guarantees under the assumption that a certain fraction of clients is honest, it follows naturally that in the presence of the sybil devices, no guarantees can be provided to the target client. This is because when the gradients are aggregated over multiple clients, and all but the target client contribute zero gradients, the aggregate solely contains this target client's gradients.

Pasquini *et al.* [144] describe a different way to circumvent SA based on the server sending out different models to different clients. While the models for non-target clients produce zero-gradients, the target client's model produces non-zero gradients which can be exploited for data reconstruction. An advantage of this method is that it does not require the server to control the client sampling or to manipulate

a fraction of clients. Note however that in their scenario, DDP can still be efficiently applied if every client adds some noise to their (potentially zero) gradients. As a consequence, the total amount of noise can be sufficient to protect the gradients of the target client. Therefore, in our attack, we rely on the controlled sybil devices to circumvent the SA.

Note also that alternative mechanisms exist to aggregate client-gradients for DDP, such as shuffling [26, 59]. However, these mechanisms can be circumvented by the exact same sybil devise-based approach.

(2) Exploiting DDP Guarantees. If DDP is in place, the gradients of the target client will be slightly noisy—even with successful circumvention of SA. However, by the design of DDP, the amount of noise added by each client is typically insufficient to provide a meaningful privacy guarantee *from the client's perspective* [99]. By meaningful privacy guarantees we mean, guarantees equivalent to what one would obtain in the LDP definition. This is in fact how DDP obtains a utility gain over LDP, which would have inserted sufficient noise to obtain per-client privacy guarantees that are independent of other clients: DDP assumes all clients will add enough noise so that the *aggregate* is sufficiently noised whereas LDP assumes each client adds enough noise to obtain *privacy in isolation*. As a consequence, in DDP, each client can add less noise locally than required for the desired total privacy level, resulting in more utility. In contrast, the guarantee provided by LDP allows the client to not trust the server or other clients. However, it comes at the expense of lower utility for the model obtained by the server.

Concretely, in an LDP version of FL, the noise added by each client depends solely on the noise scale σ and the clipping parameter c of the application. As a consequence, the local noise is sampled from a Gaussian distribution by

$$\mathcal{N}(0, \sigma^2 c^2). \tag{5.1}$$

In contrast, in DDP, the amount of noise added by each individual client additionally takes the number of clients who participate in the round into account [181]. Assuming that M clients are sampled for participation, this results in a local addition of Gaussian noise sampled from

$$\mathcal{N}\left(0, \frac{\sigma^2}{\mathsf{M}-1} c^2\right) \text{ [181]}. \tag{5.2}$$

In Figure 5.2, we present the privacy-utility trade-offs resulting from training models on the CIFAR-10 [106] dataset as a function of the total noise scale σ and the resulting models' accuracy on a test set. We train the private models with

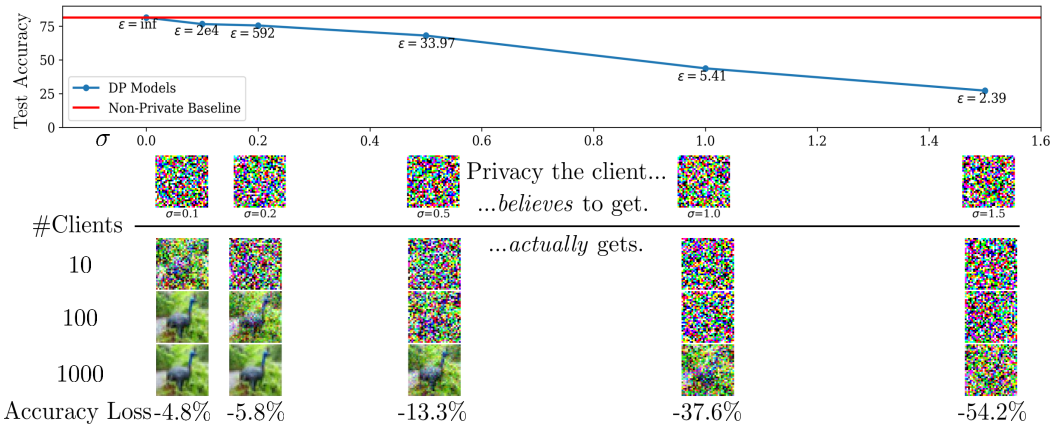


Figure 5.2.: **Privacy vs. Utility Trade-Offs under DDP.** Each point on the blue line corresponds to a model trained on CIFAR-10 with a clipping parameter $c = 1$, and a total noise scale σ. The training was conducted over 100 epochs, the resulting privacy guarantees ϵ are reported. For the non-private baseline $\epsilon = \infty$. The images depicted below the line plot display the rescaled noisy gradients of one data point of an individual client with noise calculated according to Equation (5.2) as a function of the σ, c, and the number of other clients that participate in the training round. The more clients participate, the less noise every client needs to add because, during aggregation, the total noise is determined by the sum of the individual noises. If, however, other clients do not add noise, the locally added noise is the only privacy protection every individual client has. This protection might not be sufficient. Note that the results (accuracy and reconstructed gradients) would differ if a different clip norm was used.

a state-of-the-art framework for DP-training[2] in which all hyperparameters and model architecture are tuned for the task.

Figure 5.2 provides two main insights. First and unsurprisingly, given the privacy-utility trade-offs mentioned above, the model utility decreases when the total noise scale σ increases. Second, the figure shows that the more clients participate in a given training round, the less noise each client needs to add locally. This results from Equation (5.2) which relies on the total noise being aggregated over all participating clients before sharing the aggregated gradients with the server.

[2] https://github.com/ftramer/Handcrafted-DP. Note, however, that our reported accuracy and achieved privacy levels ϵ cannot directly be compared with the values reported in the repository. This is because we use different noise scales than they do and train the model for 100 epochs while they only train for 30 epochs.

DDP assumes that each client is *honest* and adds the required noise to their gradients. However, if even one of the clients adds less than the amount of noise it should add, the desired total privacy guarantees cannot be reached. Even worse, if, as we assume for the course of our attack, a target client in FL is sampled for participation solely with controlled sybil devices that do not provide any noise for aggregation, the local noise added by the target client represents the only protection for this target client's gradients.

These results mean that there is a tension between (1) the guarantee claimed by the server (and other clients) in DDP and (2) the guarantee that a client who does not trust this server can rely on. This will lead the server optimizing for the model utility to request that clients add less noise to their gradients than what is needed for individual clients to protect their data from leaking to an untrusted server.

(3) Reconstructing Data under Noise Addition. Deciding at which point, *i.e.*, under the influence of how much noise, the reconstruction of a data point is sufficiently close to the original data point is orthogonal to this work. In particular, it will depend on the specific domain, task, and client-preference. However, due to the passive data leakage from gradients described in Section 4.2, and the increase of leakage due to our adversarial weight initialization, clients should assume that the server can extract individual data points such as the ones depicted in Figure 5.2 from their gradients. In particular, in the following experiments, we show that noise added to the client gradients is not necessarily an obstacle to the successful reconstruction of important semantics of the input.

5.3.2.2 Improving Data Reconstructing under Distributed Differential Privacy

The previous section highlights that DDP reduces to LDP with weak privacy guarantees from an individual client's perspective when other clients are untrusted with their noise addition. In this section, we show how we can even improve data reconstruction in this setup, further amplifying the small signal in the extracted gradients. We evaluate improvements for data reconstruction from noisy gradients computed under DDP on image and textual data. Whenever we report the DDP setup, we specify the respective clipping parameter c, the noise multiplier σ, and the number of participants M for the given round of the protocol. These three numbers specify how much noise each client locally adds based on Equation (5.2). The extraction of data points under FL with DDP and SA works exactly the same way as for vanilla FL (see Section 4.3). The following improvements in reconstruction solely rely on post-processing steps to reduce the effect of the noise.

Improving Image Data Reconstruction. Due to the local clipping and noise addition by the clients, the data points extracted from the gradients are not perfect reconstructions of the original data points. However, we can improve reconstruction quality by leveraging redundancy in the gradients to average out the added noise.

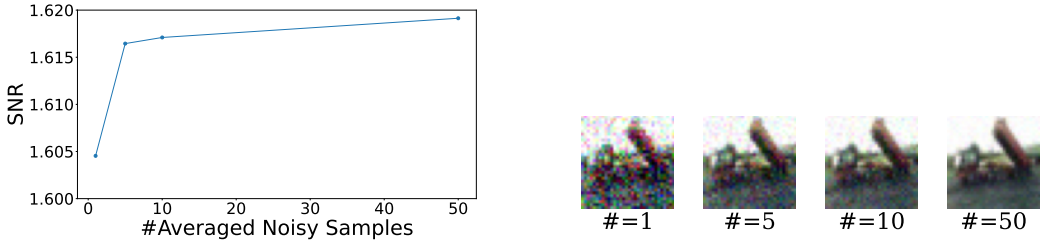


Figure 5.3.: **Effect of Redundancy for Noisy Data Reconstruction.** Mean value over #-many noisy reconstructions of the same data point (left); corresponding mean image's SNR (right). DDP setup: $c = 1$, $\sigma = 0.1$, and $M = 100$. Over an increasing number of reconstructions, the local noise averages out, yielding higher-fidelity images and increased SNR.

Figure 5.3 shows an example of the positive effect that results from averaging multiple noisy instances of the same data point: the more instances the average is calculated on, the higher the Signal-to-Noise Ratio (SNR), and the more the result resembles the original data point.

Recall from Figure 4.7 on Page 72 that our adversarial weight initialization increases natural leakage also to the point that the same data points can even individually activate more than one neuron. In this case, the data points are individually extractable from these neurons' respective gradients. As shown in Figure 4.7, we find, for example, that within a mini-batch of 100 data points from the CIFAR-10 dataset, some data points can individually activate up to 33 out of 1000 neurons.

By averaging extracted individual noisy data points and noisy overlays, the server can improve the data reconstruction fidelity and turn extracted noisy gradients, such as the ones depicted in Figure 5.4 to higher-fidelity reconstructions of the training data, for example, shown in Figure 5.5.

The server in our attack, however, without knowledge of the clients' data, has no means of determining which data points activate which neurons a priori. Therefore, it is unclear which rescaled gradients need to be averaged to improve reconstruction fidelity. To overcome this limitation, we employ similarity clustering. In this approach, the server first filters out extracted data points with a SNR below 1. This prevents too noisy instances from degrading performance. In Section 5.3.3.1 we discuss why different extracted data points have different SNRs. Then, the server runs a simple K-Means clustering on the extracted data, and finally averages all per-cluster data points. We evaluate this approach with different noise scales and mini-batch sizes B. Note that the number K of clusters has to be chosen in accordance with the mini-batch size if we want to be able to reconstruct every data point. Our evaluation suggests that clustering works best when $K \geq 2B$.

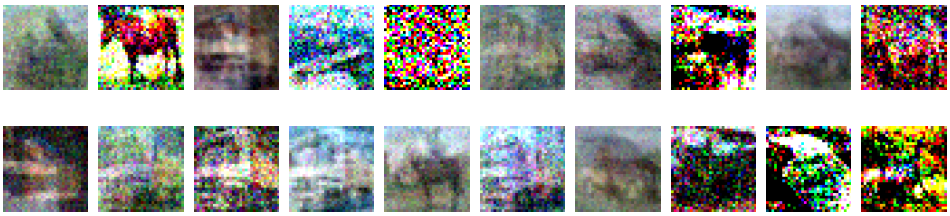


Figure 5.4.: **Rescaled Clipped and Noised Gradients.** Data extracted from a mini-batch with 20 data points from CIFAR-10 dataset, from an FC-NN with architecture specified in Table 4.1 and initialized with our adversarial weight initialization. DDP setup: $c = 1$, $\sigma = 0.1$, and $M = 100$.

In Figure 5.5, we depict the results of our clustering on data points from the CIFAR-10 dataset with a DDP setup with $c = 1$, $\sigma = 0.1$ and $M = 100$. The top row depicts 10 original data points, and the mid and bottom rows show the closest averaged clusters for mini-batches of size 10, and 20 respectively. The more instances are available for averaging, the better the resulting per-cluster averages. Please note that even when the maximum number of repetitions of an individually extractable data point was reported as 33, we average over clusters with more elements. This is due to the fact that to belong to a cluster, we do not require points to be individually extractable. The cluster can also contain extracted gradients where a few number of data points overlay but still contain semantics of the individual points.

Improving Textual Data Reconstruction. For the text classifier on IMDb, we initialize the weights of the embedding layer with a random uniform distribution (min=0.,max=1.) to create the inputs for the fully-connected layer. We then adversarially initialize this fully-connected-layer's weights with our adversarial initialization to perform extraction of the embeddings. Finally, after extracting the input's embeddings from the fully-connected layer (like they would extract an image), we map them back to the corresponding text. To reconstruct the original text tokens from a sequence of extracted embeddings, in vanilla FL, the attacker creates a lookup dictionary, mapping its initialized embeddings back to their corresponding tokens (this is the inverse mapping to the embedding layer). To avoid vector-comparisons for each lookup, we use hash values for vector embeddings as keys.

In the presence of noise introduced for DDP, the extracted embeddings are slightly noisy. To overcome this, in presence of noise we perform the lookup by searching for the token with the closest embedding measured through the ℓ_2 distance. Figure 5.6 shows the performance of a single mini-batch language extraction in presence of noise. Just as with image data, here an attacker is capable of extracting the original sentence of the clients, despite the applied noise. We do observe however that there is stochasticity involved—when parametrization does well on the data point by default, extraction gets low performance since the received gradient has an extremely low magnitude and the corresponding signal gets dominated by the

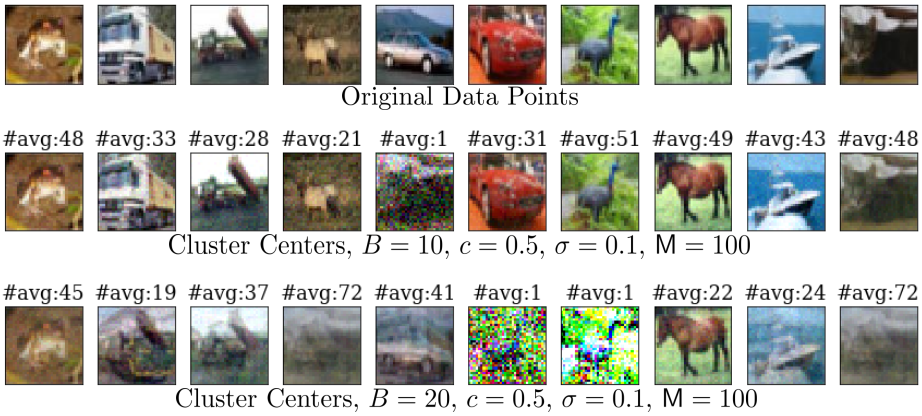


Figure 5.5.: **Original Data Points and Average Noisy Clusters.** Clusters obtained from the rescaled gradients depicted in Figure 5.4. First 10 original training data points from the CIFAR-10 dataset (top row). Averaged clusters of 10 data points reconstructed from the gradients for mini-batch size $B = 10$ (mid row), and $B = 20$ with the first 10 examples depicted (bottom row). The numbers above the images indicate how many noisy reconstructions were averaged to obtain that image.

noise. We turn to this phenomenon in the next section.

To summarize the results on image and textual data, we find that:

- Our adversarial weight initialization cause input data-diversity and redundancy in resulting gradients, which can be used to cancel out some of the applied noise.

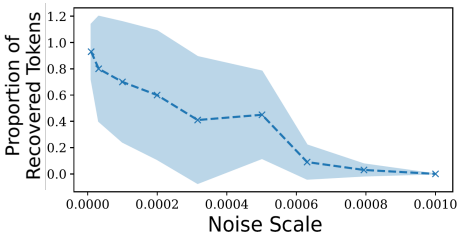


Figure 5.6.: **Extracting Text Data under DDP.** Extraction performance under noise for DDP from language model on the IMDb dataset with architecture depicted in Table 4.2. Extraction remains successful, even in presence of noise. Occasional drops in performance occur because of near-zero gradients resulting from correct data classification, *i.e.*, data points with very low loss. Error bars correspond to a single standard deviation.

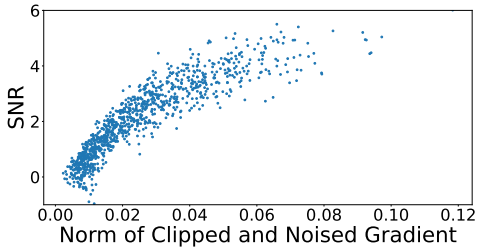


Figure 5.7.: **Gradient Norm vs. SNR in Extracted Data.** Norm of the clipped and noised gradients of 1000 weight rows against SNR in corresponding extracted data point, *i.e.*, the rescaled gradients. With higher gradient norms, the SNR in the extracted data increases. DDP setup: $c = 1$, $\sigma = 0.1$, and $\mathsf{M} = 100$.

- NLP is not safe from attacks described in this paper, despite a more sophisticated input-embeddings mapping.
- Despite using DDP, an attacker often can reconstruct semantic information on the individual client data points. This is because, in the presence of untrusted other clients, DDP reduces to LDP with weak privacy guarantees from the perspective of an individual client.
- As shown in Figure 5.2, having clients add more noise locally comes with a significant decrease in utility which makes the solution less practical.

5.3.3 Adversarial Model Manipulations for Data Extraction under Distributed Differential Privacy

Throughout our experiments, we observe that with the exact same scale of noise added to all gradients, some extracted data points have a significantly higher SNR than others. This effect translates into different levels of semantic similarity in the extracted data with respect to the to original data as we show in Figure 5.4. In this section, we explain this observation and sketch how it can be leveraged to better extract data in the presence of noise.

5.3.3.1 Influence of the Gradient Norm on Extraction Success

We find that the SNR of an extracted data point is tightly bound to the magnitude, *i.e.*, the norm, of the respective gradients. In Figure 5.7, we depict the SNR in rescaled clipped and noised gradients, *i.e.*, the extracted data points, against their respective gradient norms. It shows that with higher magnitude gradients, the same amount of noise has less impact on the signal, whereas, with smaller magnitude gradients, the same amount of noise largely dominates the signal. Therefore, the increased magnitude of model gradients results in increased data leakage.

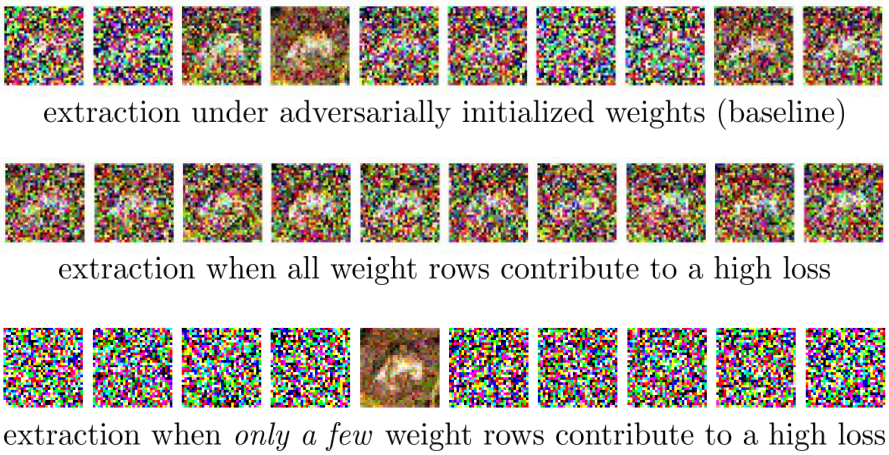


Figure 5.8.: **Extraction Success under Model Manipulations to Circumvent DDP.** Under adversarial weight initialization only, gradients at different weight rows have varying SNRs under the same amount of added noise, depending on their magnitudes (top). When the shared model is further manipulated and all weight rows contribute equally to a high loss, their gradients will be clipped, which results in equal information loss for all of them (middle). When only a few weight rows contribute to a high loss, theirrespective gradients can keep a high magnitude over clipping (and subsequent noise addition), which allows for higher fidelity extraction. DDP setup: $c = 1$, $\sigma = 0.1$, and $\mathsf{M} = 10$.

The norm of a weight row's gradients in the model depends on the model's loss. In general, higher loss results in higher magnitude gradients, in particular for the weight rows that most contribute to the loss. Intuitively, to increase data leakage from noisy gradients, the server could, therefore, manipulate the shared model to produce a higher loss. In the best case, the loss would be caused by all weight rows in the fully-connected layer used for extraction. This ensures high-magnitude gradients at all weight rows for enhanced extraction at all of them.

However, in DDP, before noise addition, clients perform a clipping step, bounding the maximum signal in a gradient update. Clipping limits the total norm of a model layer's gradients to the clipping parameter c. If all weight rows have high gradients, their joint norm will exceed c, and therefore, all of them will have to be clipped. This results in a loss of semantic information in all the rescaled gradients, *i.e.*, the extracted data points, see the middle row in Figure 5.8.

5.3.3.2 Adversarial Model Manipulation for Improved Data Extraction

As a solution to overcoming the loss of semantic information induced by the clipping, we propose a novel adversarial model manipulation.

Our construction has the two following effects:

1. It causes a high loss in the model to increase the gradients' magnitude.
2. It "attributes" this loss to only a few weight rows, such that only their gradients obtain a high magnitude. This circumvents the negative effect of clipping for them because the *overall norm* of the layer's gradients will stay below the clipping parameter c. As a consequence, the high magnitude gradients of the weight rows with attributed loss can be used for higher fidelity data extraction, see the bottom row in Figure 5.8.

We sketch the intuition on how to design this adversarial model manipulation for an FC-NN with two layers: the first layer is initialized with our adversarial weight initialization for extraction, the second, *i.e.* the classification layer, is modified to obtain the desired effect of the loss. To cause a high loss in the model without any knowledge on the client data, we add an additional neuron to the classification layer, *i.e.*, an additional class that does not occur in the data distribution. Then, we set most of the weights that connect the output from the previous layers' neurons to this additional class to very small values, *e.g.* zero, and the weights for a few neurons' output to high values, *e.g.* one. Note that the first fully-connected layer uses a ReLU activation function. Hence, all its output values will be either zero or positive. By multiplying them with high positive values for the added class in the second fully-connected layer, we cause the data points to be misclassified into this class with a relatively high loss even without knowledge on the concrete training data by the server.

Figure 5.9 visualizes this idea. In the figure, the gradients for adversarial weight row 2 will be large, whereas the gradients at adversarial weight row 1 and n will be zero. As a consequence, the rescaled gradients of these latter two rows consist purely of noise. However, the overall gradient magnitude will be below c, and therefore, the gradients at adversarial weight row 2 enable high-fidelity extraction, see the bottom row in Figure 5.8.

We experimented with different fractions of weight rows that contribute a lot to the high loss by setting different amounts of weights that connect to the added class to *one* while setting the rest of the weights to *zero*. The resulting SNRs are depicted in Figure 5.10. It shows that when all weight rows contribute equally to the high loss, see Figure 5.10 (d), their rescaled gradients have a similar SNR. This is due to all of them being equally affected by the clipping. However, when fewer weight rows contribute to the high loss, their respective rescaled gradients, *i.e.*, a few extracted data points, have a higher SNR, and thereby allow for higher fidelity extraction.

This two-layer construction naturally integrates with other architectures starting with convolutional and embedding layers described in this work. We leave fine-tuning and the extension of our construction to architectures that end with more than two fully-connected layers for future work. However, the approach highlights

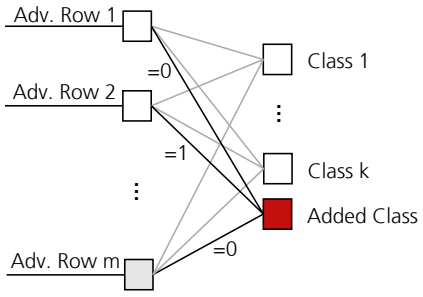


Figure 5.9.: **Adversarial Model Manipulation for Extraction under DDP.** A sketch of an adversarial model manipulation that circumvents the negative effect of the clipping in Differentially Private Stochastic Gradient Descent (DPSGD) on data reconstruction fidelity.

that even when DDP is in place, an untrusted server can maliciously initialize a model to increase the likelihood of reconstructing points with high-fidelity.

5.3.4 Conclusion

In summary, the assessment of trust assumptions and privacy in hardened FL highlights that even when implementing the so-far, known as most protective extensions of FL, the server still holds an upper hand on information leakage in the protocol. Therefore, even these hardened extensions of FL cannot provide privacy against an untrusted server.

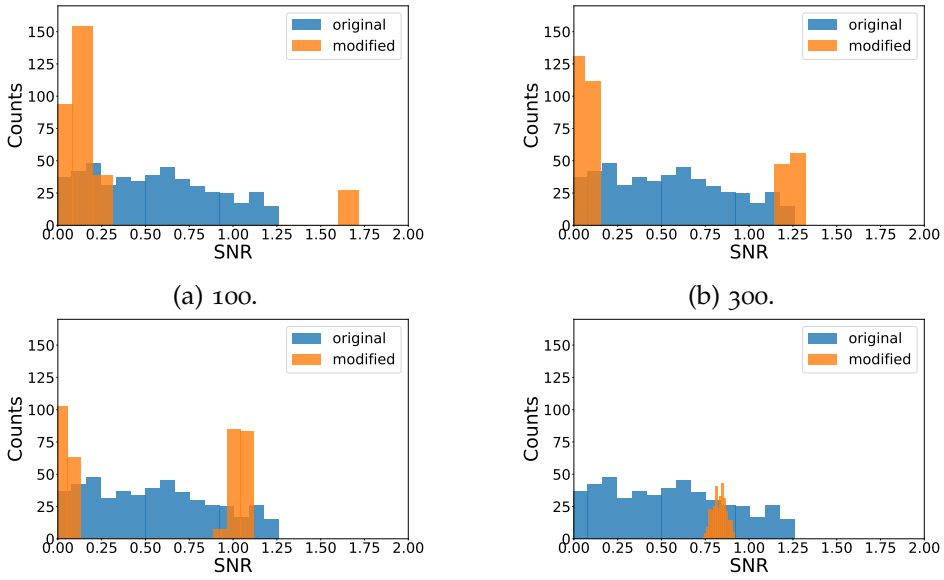
5.4 Towards Relieving Trust for Privacy-Preserving Federated Learning

Based on our study of attack vectors, trust assumptions, and their respective impact on practical privacy guarantees, in this section, we give recommendations to guide future research and implementation towards private FL. Therefore, we first distill sources of privacy leakage in the current practical FL deployment. Then, we propose strategies to mitigate privacy risks and provide recommendations for clients participating in the protocol.

5.4.1 Identifying Privacy Problems in Federated Learning

Our investigation of different FL deployments and their respective attack vectors highlights the following sources of vulnerability to privacy leakage in the protocol:

1. The server's ability to control a fraction of clients.
2. The server's ability to sample clients for participation in each protocol-round.



(a) 100. (b) 300.

(c) 500. (d) 1000.

Figure 5.10.: **SNRs of Rescaled Clipped and Noised Gradients,** *i.e.*, the extracted data points. The first fully-connected layer used for extraction consists of 1000 neurons. The respective weight rows are initialized with our adversarial weight initialization. The second layer is manipulated by adding an additional class neuron and setting {100, 300, 500, 1000} of the 1000 weights that are connected to this neuron to one. The remaining weights that go to this neuron are set to zero. The original baseline consists of randomly initialized weights for the second fully-connected model layer. $c = 1$, $\sigma = 0.001$, and $M = 1$.

3. The difficulties of integrating DP for the right trust assumptions and with reasonable privacy-utility trade-offs.
4. The server's ability to manipulate the model.
5. The clients' inability to verify each other's correctness.
6. The clients' inability to verify and validate model updates.

While points (1) and (2) enable the server to circumvent SA, point (3)—in combination with (1) and (2)—enables to extract private information even if DDP is implemented. Point (4) enables the server to choose model architectures that facilitate data extraction and to adversarially (re-)initialize the model weights to increase the percentage of extractable data or extract from a wider range of model architectures. The points (5) and (6) highlight that clients have to either trust the server, or rely on cryptographic protocols that take responsibility away from it. In particular (5) represents an issue for the correct execution of DDP, while (6) impedes clients in detecting active attacks against the model weights.

In the following sections, we elaborate on these points and discuss methods to reduce the associated vulnerabilities.

5.4.2 Detection of Client Manipulation

For this aspect, it comes down to the question: *Can a client detect if another client is manipulated?* This question is of high importance since a client should not participate in the SA for DDP if other clients are manipulated. This is due to DDP only providing meaningful privacy guarantees by aggregating all clients' noisy gradients. As a consequence, even if only a small fraction of clients is manipulated and does not contribute noise, the privacy of all participating clients degrades.

The issue of detecting manipulated clients addresses points (1) and (5), since, if clients were able to verify the correctness of other clients' gradients, the server could no longer exploit its ability to control a fraction of clients to circumvent SA and DDP. The main issue for verification is that in FL, clients usually do not have direct communication channels with each other, but rely on the server [144]. Therefore, when the server is untrusted, clients cannot verify that other clients are not manipulated.

There exist, however, several methods that are, in principle, adaptable to detect manipulated client gradients. Weng *et al*. [194] propose to use Zero Knowledge (ZK) proofs to verify private benchmarks. In their scenario, a public ML model is evaluated on private data. Therefore, the owner of the private data publicly commits to the data and then locally evaluates the accuracy of a public model. The data owner can use the ZK protocol to prove that the public model was executed on the committed data. In FL, the shared model is known to all the clients, however, their local data has to remain private. As an adaptation, instead of proving the accuracy of the shared model on private data, a given client could prove that a gradient was computed for the shared model on the local private data. This approach limits the attack space to only the first step of the protocol when a malicious client can commit to manipulated data. The drawback of this method lies in its high computational cost: The verification of accuracy requires fewer operations than the verification of gradients, yet, already the reported execution time of accuracy verification in [194] on 100 test images is 8.2 minutes for LeNet5, 4.4 hours for ResNet50, and 7.3 hours for ResNet101.

Biscotti [166] is a block-chain based FL system that also includes the verification of clients' updates via Krum [27]. This verification approach is strictly weaker in terms of guarantees than via ZK proofs but more practical. First, Biscotti selects clients as verifiers in a randomized way to ensure that malicious clients cannot deterministically choose themselves to verify a victim's gradient. Second, pre-committed noise for DP is added to the clients gradients, thus masking them before they are sent to a verifier. Finally, the Biscotti system opts for aggregation of

unnoised model updates since by aggregating differentially private updates, the final model has lower utility. This design choice makes the system vulnerable to privacy leakage from clients' gradients.

Yet, at least in the FL deployment targeted in this work, clients cannot freely decide on their participation in FL training iterations. Instead, the server controls the client sampling, and once a client is selected for participation, it is assumed to contribute to the training. Hence, even if the clients were able to identify manipulated other clients, in this deployment, they could still not refrain from participation.

5.4.3 Control on the Client Sampling Mechanism

The previous section highlights the importance of the sampling mechanism to client privacy, see point (2). To address this point, mechanisms where the clients perform a self-sampling, such as [73], have been put forward. These allow clients to decide about their participation and, thereby, reduce the power of the server.

Other adaptations of FL, such as anarchic FL [198] go even further. Not only do they allow clients to decide in which iterations of the FL protocol they want to participate, they also allow the clients to perform training with parameters fully according to their preferences. In this setup, the server can only sample from clients that agree to participate in a given iteration. Therefore, the shared model is published and clients pull it at a given state with an associated time-stamp. Then, clients perform the local training and share the resulting model updates together with their time stamp of the shared model.

Also in Biscotti [166], no server is in charge of the client sampling, instead, clients are weighed by the value, or stake, that they provide to the system. Then, Biscotti uses consistent hashing in combination with verifiable random functions to select key roles for clients who coordinate the privacy and security of model updates. Such adaptations allow to mitigate risks that stem from the server's control on the client sampling mechanism.

5.4.4 Implementations of Differential Privacy

As we described in Section 2.3.5.3, next to DDP, there are LDP and CDP which allow to integrate DP into FL. When it comes to LDP, related work agrees that the amount of noise that is introduced through it, yields poor model utility, *e.g.* [46]. When it comes to CDP, assuming an untrusted server, the approach cannot yield meaningful privacy guarantees since the server could simply skip noise addition or extract the data before adding noise.[3]

[3] Yet, standard systems, such as the one described in [121] seem to be implemented that way and state: "We can provide accurate estimates of cumulative sums with a strong DP guarantee by utilizing negatively correlated noise, added by the aggregating server".

Note: For a secure aggregation of sensitive statistics (instead of ML model gradients), there exist solutions of CDP without a trusted aggregator [156, 157]. Such improved aggregation protocols that achieve a correct joint noise addition would also solve the point (3), given that sampling is not manipulated. This joint noise addition could theoretically be implemented by full SMPC.

5.4.5 Secure Aggregation and Secure Multiparty Computation

SA in FL can be eluded [144], or circumvented as described in Section 5.3.2, even though the cryptographic protocol is correct. This does not only override the advantages of aggregation, such as the protection of provenance of the respective gradients, but also degrades the privacy guarantees of mechanisms that are built on top of it, such as DDP. A strategy to implement meaningful privacy guarantees to address point (3) would lie in replacing SA with a full SMCP protocol between the clients inside of which a sufficient amount of noise is always added.

However, both SA, as well as other SMPC protocols introduce communication, computation, and storage costs in FL. The assessment of costs incurred for clients though SA, highlights (1) communication and computation costs for key agreement between clients, (2) communication costs of exchanging the actual messages, and (3) additional storage [33]. For (1), cost complexity is $O(n^2 + nm)$, with n indicating the number of clients, and m the length of input they hold. For (2), the costs can be assessed by $O(n + m)$, with possible improvements [45]. Since clients require to store each others' keys and secret shares, their storage requirements can be assessed as $O(n + m)$ [33]. Note also that the SA protocol [33] is based on two strong assumptions, namely the availability of a Public Key Infrastructure (otherwise, clients have to trust the server to honestly forward their public keys to all other clients for verification), and on the availability of a shared seed for a pseudorandom generator. Without this seed—-usually computed through a key agreement by the users—the communication costs for the clients increase significantly. Also note that SA operates over finite fields which requires different computations than the ones on the original client gradients, which are usually floating point values. For the server, the respective costs are (1) $O(mn^2)$, (2) $O(n^2 + mn)$, and (3) $O(n^2 + m)$ [33]. Notably, several costs in SA grow quadratically with the number of clients, which limits the number of clients that can participate in the protocol without significant losses in efficiency.

To the best of our knowledge, so far, no protocol for jointly adding sufficient amounts of noise to client gradients in SMPC exists. Hence, we cannot provide a concrete cost assessment here similar to the one for SA. Given, however, the gradients' high-dimensionality, the communication costs of any such approach are expected to exceed the costs of SA by far, and thereby, seem to be prohibitive for practical application.

What is more are the implicit costs of introducing SA and SMPC into FL: given their communication costs, both approaches effectively reduce the number of clients per training iteration of the protocol to several hundreds [32]. Therefore, positive effects such as the *linear speedup* (*e.g.* [201]), *i.e.*, the fact that the convergence time in FL decreases linearly as the number of clients increases, cannot be leveraged.

5.4.6 Control on the Shared Model

To address points (4) and (6), we focus on the shared model. There are two approaches how clients can protect their sensitive data against model manipulation. The first approach relies on detecting these manipulations and then refraining from participation in the protocol (assuming that clients have this possibility, see Section 5.4.3). The second relies on preventing model manipulations in the first place.

Detectability of our Adversarial Initialization. Our adversarial weight initialization contains a few characteristic elements, *e.g.* slightly larger negative than positive weights. A difficulty in detection of our adversarial initialization lies in distinguishing whether they are the product of our adversarial weight initialization, or of the previous training. Also, access to several model weights over time does not ease detection since it is impossible for a client to decide whether the newly received model weights are the result of a legitimate optimization step from their previously received ones. This is due to several reasons: (1) The clients have no insights into the data of other clients. (2) Even if they had, ML training algorithms rely on stochastic elements. These combined with the non-determinism of modern hardware, make it difficult to reproduce training runs [95]. (3) This is aggravated by the fact that in large-scale FL applications with control flow mechanisms, such as *Pace Steering* [32] clients are not sampled at every round of the protocol [152]. Hence, their new model weights usually result from several optimization steps.

In addition, a client can also observe the model *functionality* for detection of our adversarial weight initialization. Note that our OM server only sends out adversarially initialized models during short periods of time, and the benign model otherwise. When clients receive an adversarially initialized model before the benign model, they cannot tell whether the high loss values are due to the model being in an early stage of training. Even when clients first receive the benign model and then an adversarially initialized one, they still are lacking insights into the other clients' data and possible distribution shifts within them which can lead to decreases in model utility for their own data. Also, research has shown that for clients whose data stems from the tails of the data distribution, FL does not necessarily lead to an improvement of the model on their data [202]. Hence, clients cannot, with high fidelity, attribute the increased loss to our adversarial initialization.

Finally, we would like to highlight that all possible detection mechanisms assume that clients have access to the shared model and the gradients computed on their data. Latest FL applications are encapsulated in dedicated software partitions [191] and it is unspecified what the clients can explicitly access. Hence, the server deploying these FL applications on the clients' end should be in charge to implement such detection mechanisms to prevent privacy breaches that can occur due to, for example, malicious employees. However, then again the clients have to trust the server to do that correctly.

Secure Deployment of the Shared Model. Given the difficulty for clients to detect manipulations in the shared model, preventing these manipulations in the first place presents itself as a promising solution.

First of all, we note that the chosen model architecture has an impact on the number of required manipulations from the server. While architectures that start with a fully-connected layer allow for direct data extraction, in architectures that start with convolutional layers, additional manipulations are required to forward the client data to the first fully-connected layer for extraction, see Section 4.4. Additionally, we show that lossy layers, such as pooling and dropout, at least reduce the quality of extracted client data as shown in Section 4.5. Therefore, architectures containing this type of layers should be preferred when specifying the shared model.

Once the architecture is determined, when it comes to deployment, the shared model can be deployed in a Trusted Execution Environment (TEE), such that one can obtain attestations that the model was initialized with a standard initialization (in contrast to being initialized, for example, with our adversarial weight initialization). Furthermore, applying model updates in the TEE can prevent subsequent model manipulations, such as malicious re-initialization of the model weights. To make sure that clients always receive this well-controlled and not manipulated model, we, furthermore, suggest releasing the shared model publicly, for example, in a block-chain. This makes it impossible to manipulate and change a shared model after release. A drawback of this solution is that it offers outside attackers access to several intermediate model states. We argue that this is not too restricting, though, since the shared model is also sent out to a few hundreds or thousands of clients during each round. Hence, there exists the possibility of the internal model states being leaked, anyways.

5.4.7 Recommendations for Clients

Finally, we also want to briefly address the question of *What can each individual client do to limit data leakage from its shared gradients in presence of an untrusted server?* Based on studying the factors that influence the success of our attack, we can give the two following main recommendations: (1) Clients should, if they are able to specify within the FL protocol, calculate their gradients over large mini-batches of data

since this decreases the extraction success of our adversarial weight initialization. If possible, they should average their gradients locally or perform several iterations of local updates for increased protection. (2) Clients should add enough noise to their local gradients to benefit from meaningful DP-guarantees. They should not rely on other clients to provide additional protection, as we have shown in Section 5.3.2. Yet, this latter point might come at intolerable utility loss for the server.

5.5 Conclusion

Truly privacy-preserving ML must defend itself from attackers that are malicious. In this chapter, we analyzed the question of the minimum trust model that is required to obtain meaningful privacy guarantees for clients in FL. We showed that existing FL deployments, even the hardened ones that implement dedicated privacy-protection can only provide privacy guarantees if the clients trust the server. Otherwise, the protocol requires adequate additional protection methods, such as the application of cryptographic protocols that perform the noise addition without relying on trust in other clients or the server. Indeed, most of the methods for protection aim at shifting power from the server to the conglomerate of clients and come with significant costs or overhead. Therefore, such systems are not yet practically in place. As a consequence, as of yet, we recommend clients to only participate in FL protocols that are orchestrated by a trusted server.

Chapter 6

Impact of Differentially Private Training on Model Robustness

While the previous two chapters have mainly approached the question of how attacks can be used to target the privacy of Machine Learning (ML) models, this chapter aims at studying the effect of attacks on models trained with privacy guarantees. More concretely, it presents a study on the effect of training ML models with Differentially Private Stochastic Gradient Descent (DPSGD) on the resulting models' vulnerability against adversarial examples [78, 92, 115].

Adversarial examples are data points that contain small and human-imperceptible perturbations, forcing the attacked ML models into misclassifications. Such misclassification can have a high impact on the security of the application that the ML model is applied for as we have already briefly discussed in Section 2.3.2.

Even though the creation of models that are both private and secure at the same time poses a desirable goal, the majority of previous work focused on either one of the tasks. Only recently has the intersection of the different research branches received more attention [93, 146, 147, 150]. The interrelation between security and privacy can be approached from two sides: either by evaluating the privacy implications of making a model more robust or by studying how private training impacts model robustness. So far, the former, namely the question of the influence of adversarial retraining—a common method for increasing model robustness—on model privacy, has been investigated more thoroughly [72, 88]. It was shown that adversarial retraining decreases the membership privacy of an ML model's training data points. *I.e.*, through adversarial retraining, for an attacker, it becomes easier to determine whether a specific data point was used during training [87, 124, 171]. Regarding the latter perspective, first, results [183] suggest that applying DPSGD to train an ML model has negative impacts on the model's robustness, even though there might potentially be privacy parameter combinations that are beneficial for both goals. However, the experiments conducted to evaluate robustness in [183] are limited to one specific kind of method to generate adversarial examples, namely

methods that rely on the model gradients (called *gradient-based attacks* [78, 115]). Furthermore, their study only considers one type of model. Hence, it remains unclear whether their findings on the effect of training with privacy on model robustness can be generalized.

However, since ML models increasingly operate in sensitive areas, such as health care, these models must implement both privacy for their training data and robustness against targeted attacks. By thoroughly studying the effects of training with DPSGD on model robustness, we hope to better characterize existing trade-offs between both goals and to inform the choice of parameters that help implement both at the same time.

Therefore, to further shed light on the intersection between ML models' robustness and their privacy, this chapter presents a comprehensive study building upon current findings. We start by a more thorough introduction on adversarial examples to extend the brief overview in Section 2.3.2. In particular, we describe the methods used to generate adversarial examples in the context of this work. We then introduce our experimental setup and resulting findings on the impact of DPSGD training on robustness. We evaluate gradient-based [78, 115], gradient-free [34] and optimization-based methods [42] for adversarial example generation. See the following sections for a more thorough introduction to the different methods. Furthermore, we study adversarial transferability [78] among models with different privacy settings, and between private and non-private models. Our experiments confirm prior findings [183] that training with privacy guarantees impacts model robustness. We end this chapter with a discussion of possible reasons for this impact and an outlook on how to optimize for security and privacy at the same time.

The results presented in this chapter have been made available online as a preprint [31] prior to writing this dissertation. As a consequence, this chapter contains material adapted from [31].

6.1 Background and Related Work

This section first formally introduces the concept of adversarial examples, presents methods to generate them, and discusses the principle of adversarial transferability. Finally, it provides an overview of related work studying the intersection between privacy and ML model robustness.

6.1.1 Adversarial Examples

According to Szegedy *et al.* [179], adversarial examples can be characterized as follows: Given an original data point x, the point x' is an adversarial example if

1. $x' = x + r$ for a small perturbation r with $||r|| \leq \beta$, and

2. $f_{\mathcal{W}}(x) \neq f_{\mathcal{W}}(x')$, with $f_{\mathcal{W}}$ being the ML classifier under attack.

Hence, while x' differs only slightly from x, the attacked model wrongly predicts the target class of x'. See Figure 2.2 on page 12 for a visualization of the concept. Note that in literature, what we refer to as β is often denoted by ε. Since we already use ε to refer to the privacy budget in Differential Privacy (DP) (see Section 2.3.5), we deviate from this naming convention to avoid confusion.

Using the notation introduced in Section 2.1, we can formalize the generation of adversarial examples as a constrained optimization problem [179]: We aim at minimizing the distance $d_p(x, x')$ with $x' = x + r$, such that $f_{\mathcal{W}}(x') = t$ for a chosen target class t (**constraint 1**) with $x' \in [0, 1]^m$ (**constraint 2**). While constraint 1 makes sure that the produced adversarial example is indeed misclassified, constraint 2 ensures that x' is a valid data point, *i.e.*, it has the normalized dimensions of x.

Measuring Adversarial Robustness. The distance between a benign data point x and its adversarial counterpart x' is usually measured using an ℓ_p-norm. This distance is important to judge an ML model's robustness. A model that already misclassifies data points under the presence of smaller adversarial perturbations is considered less robust than a model which only misclassifies data points under the presence of larger perturbations. In addition to the amount of perturbations, for methods that create adversarial examples using multiple optimization steps, also the number of steps required until the model starts misclassifying the data point is used to evaluate model robustness. A model for which successful adversarial example generation with a given maximum level of perturbation β requires fewer optimization steps is considered less robust than a model for which more optimization steps are required.

Generating Adversarial Examples. In literature, several methods have been proposed to generate adversarial examples, which can be divided in gradient, optimization, and decision-based methods [204]. A summary of current attack methods can be found in the survey by Ren *et al.* [155]. In the following, four widely used methods that were included in the experiments of this work are introduced. Other important methods worth mentioning are DeepFool [127] and the One-Pixel-Attack [175]. The former method explores and approximates the decision boundary of the attacked model by iteratively perturbing the inputs. For the latter one which operates on vision data, solely one pixel of a benign image is changed to generate the adversarial example.

6.1.1.1 Fast Gradient Sign Method

The Fast Gradient Sign Method (FGSM) [78] is a single-step, gradient-based method that relies on backpropagation to find adversarial examples. More in detail, the method takes an input data point x, obtains the ML model's prediction on the data point, and then calculates the loss w.r.t. to correct target class y. Based on the

loss, the gradients are calculated and the sign of these gradients is determined. Finally, based on these signed gradients, the adversarial example is calculated. For the model parameters $\mathcal{W}$, input x, target y, and loss function $\mathcal{L}(\mathcal{W}, x, y)$, x' is calculated as follows

$$x' = x + \beta \operatorname{sign}(\nabla_x \mathcal{L}(\mathcal{W}, x, y)). \tag{6.1}$$

Multiplying the result with a small value β aims at making the perturbation imperceptible by a human-observer but large enough to cause the model to misclassify the example. However, the method does not offer a guarantee that, after adding the calculated perturbation to x, the resulting x' will look similar, instead it is a trade-off between making the model misclassify the instances and creating non-noticeable perturbations [78].

6.1.1.2 Projected Gradient Descent

The Projected Gradient Descent (PGD) method [115] represents a multi-step variant of FGSM. It serves to find better adversarial examples, *i.e.*, adversarial examples that are less detectable by a human-observer. Given a distance norm p, an input x'_0 is initialized within an β ball in ℓ_p around the original data point. This means that the distance between x and x'_0 does not exceed a threshold value β in ℓ_p. The adversarial example x'_0 is then iteratively adapted with FGSM, and if necessary, the perturbation is projected back into the β ball around x to make sure, x' stays close enough to x. Starting with x'_0, a small step size α, and projection Π_{ℓ_p}, the adversarial example x'_t at iteration t is constructed as follows

$$x'_t = \Pi_{\ell_p}(x'_{t-1} + \alpha \operatorname{sign}(\nabla_x \mathcal{L}(\mathcal{W}, x'_{t-1}, y)). \tag{6.2}$$

6.1.1.3 Carlini and Wagner

The optimization-based Carlini and Wagner (CW_2) method [42] specifies a loss function based on how close the model prediction is to the target class t. The authors found that the best loss function to be used for optimization is given by

$$\mathcal{L}(\mathcal{W}, x', t) = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -d). \tag{6.3}$$

The variable Z represents the *logits* of the target model $f_{\mathcal{W}}$, *i.e.*, the non-normalized prediction probability vectors over all output classes of $f_{\mathcal{W}}$. Then, $\max\{Z(x')_i : i \neq t\}$ is the highest prediction probability over all classes that are not the target class t. $Z(x')_t$ represents the prediction probability for the target class t, and $\max\{Z(x')_i : i \neq t\} - Z(x')_t$, therefore, is the difference between the current model's highest class prediction and what we want the model to predict. The term $-d$ within the outer max value specifies a lower limit on the loss. As a consequence,

d controls the confidence which we require from the model in the prediction of the adversarial example.

Based on this loss function, the constraint for the adversarial example generation in the CW_2 method can be formulated by

$$\min \parallel x' - x \parallel_2^2 + c \cdot \mathcal{L}(\mathcal{W}, x', t), \tag{6.4}$$

such that $x' \in [0,1]^m$. The constant $c > 0$ is determined by binary search to find the smallest c for which misclassification occurs. Moving the difficult part of the optimization problem into the min-function, as done in this method, is an approach used to make optimization problems easier to calculate, which the CW_2 method then does to determine adversarial examples over multiple computation steps.

6.1.1.4 Boundary Attack

The Boundary Attack (BA_2) [34] is a multi-step gradient-free method that relies solely on the model's decision to generate adversarial examples. Therefore, x' is initialized as an adversarial example, such that, $f(x) \neq f(x')$. Afterwards, a random walk on the boundary between the adversarial and the non-adversarial region is performed in order to minimize the distance between x and x', while keeping x' adversarial. With the BA_2 method, adversaries are able to craft adversarial examples even if the gradients of the attacked Neural Network (NN) are not available.

6.1.2 Adversarial Transferability

It has been shown that adversarial examples transfer between models [78], *i.e.*, adversarial examples that are generated on one model are often successful in fooling other similar models. The degree of success depends on several factors, among which the complexity of the model that the adversarial examples are crafted on, and the similarity between both models [52].

6.1.3 Protecting Against Adversarial Examples

Currently, adversarial training is considered the most effective method to protect against such attacks [155]. During this computationally expensive process, adversarial examples are included in the ML model training procedure, using their original labels [115]. Thereby, the model learns to still predict the perturbed data points into their correct original class.

6.1.4 Related Work on Privacy vs. Robustness

Research suggests that the goals of achieving privacy and robustness in ML models are not always in line. It was found that adversarial training, *i.e.*, training an ML model with adversarial examples, increases membership privacy risks [124, 171].

LeNet architecture	Custom architecture
Conv(f=6, k=(3,3), s=1, p=valid, act=relu)	Conv(f=16, k=(8,8), s=2, p=same, act=relu)
2D Average Pooling(pool size=(2,2), s=1, p=valid)	2D Max Pooling(pool size=(2,2), s=1, p=valid)
Conv(f=16, k=(3,3), s=1, p=valid, act=relu)	Conv(f=32, k=(4,4), s=2, p=valid, act=relu)
2D Average Pooling(pool size=(2,2), s=1, p=valid)	2D Max Pooling(pool size=(2,2), s=1, p=valid)
Flatten	Flatten
Dense(n=120, act=relu)	Dense(n=32, act=relu)
Dense(n=84, act=relu)	Dense(n=10, act=None)
Dense(n=10, act=None)	

Table 6.1.: **ML Model Architectures.** Architectures of the models used in the experiments. f: number of filters, k: kernel size, s: stride, p: padding act: activation function, n: number of neurons [31].

With decreased membership privacy, it might be easier for an attacker to determine whether a specific data point has been used for model training [168]. This negative impact of adversarial training on models' privacy can be explained by overfitting since the training can enforce the training data points more strongly to the model [87]. Thereby, their membership becomes more easily determinable [199].

Examined from the opposite perspective, it was shown that the noise of DP training can be exploited to craft adversarial examples more successfully [72]. In the work that is closest to this one, Tursynbek *et al*. [183] observed that DP training can have a negative influence on adversarial robustness. Yet, the authors identified some DP parameters that give the impression of improved robustness against gradient-based adversarial example crafting methods.

Some research has already been dedicated to aligning model privacy and robustness. Pinot *et al*. [150] showed that model robustness and DP share similarities in their goals. Furthermore, several mechanisms to integrate (provable) adversarial robustness into DP training have been proposed [93, 146, 147].

6.2 Experimental Evaluation

This section describes our robustness evaluation for models trained with different DPSGD parameters. We start by introducing our method and experimental setup. We then move on to evaluating different types of attacks. Finally, we analyze adversarial transferability.

6.2.1 Method and Experimental Setup

We aim to explore the intersection between privacy and security in NNs. Therefore, more specifically, we trained ML models using the DPSGD with different sets of parameters and then measured the resulting models' robustness against adversarial examples. We varied both the noise scale σ, and the clipping parameter c for

the DPSGD training. In accordance to [183], the value ranges were set to $\sigma \in \{0, 1.3, 2, 3\}$ and $c \in \{1, 3, 5, 10\}$.

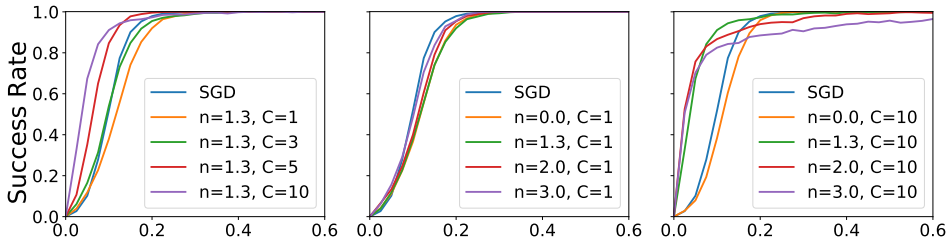
To evaluate robustness, we relied on adversarial examples generated by three different attack methods. The PGD_∞, CW_2, and BA_2 were used as examples of a multi-step gradient-based, multi-step optimization-based, and a multi-step gradient-free attack, respectively. The experiments in this paper were conducted using the MNIST dataset, and the respective adversarial examples were generated using the Foolbox framework [153]. For every experiment, 1000 adversarial examples were generated based on 1000 random test data points that were predicted correctly by the model under attack.

As can be seen in previous guidelines on the evaluation of adversarial robustness [9, 41], the selection of attack methods heavily influences the perception of robustness. Therefore, to obtain more general results, for our experimental evaluation, we used two different ML model architectures, an adaptation of the *LeNet* architecture [109] (1), and a *custom* conv-net architecture (2), depicted in Table 6.1. The ML models are implemented using TensorFlow [1] version 2.4.1 and trained for 50 epochs using a learning rate of 0.05, and a mini-batch size of 250. Stochastic Gradient Descent (SGD) and TensorFlow Privacy's [66] implementation of the DPSGD were used as optimizers to train the models without and with privacy, respectively.

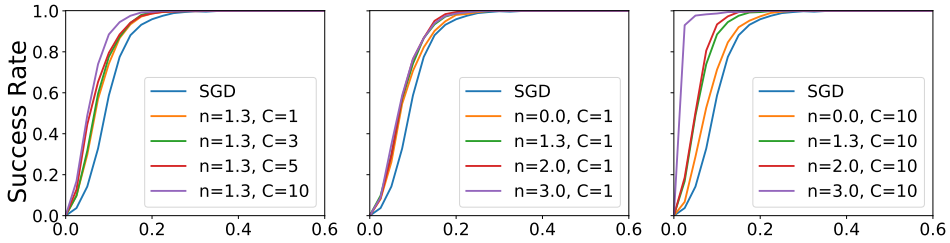
6.2.2 Robustness Evaluation with the PGD_∞ Attack

Perturbation-based Analysis. In the first experiment, both the custom and the LeNet model were attacked using adversarial examples generated by PGD_∞. The robustness was quantified using the adversarial success rate, *i.e.*, the percentage of generated adversarial examples that successfully cause misclassification of the model. As PGD_∞ creates bounded adversarial examples, we evaluated the success rate for different maximum magnitudes of adversarial perturbations β, *i.e.*, maximum distances between the original data point and the resulting adversarial example. The perturbation budget β was increased successively from 0.0 to 0.5 in steps of size 0.025 for a fixed number of 40 attack iterations. At every perturbation value, the adversarial success rate was measured. Higher success rates for the same perturbation budget suggest lower model robustness. See Figure 6.1 for the results. For both model architectures and all privacy parameter combinations, an increase of the perturbation budget β resulted in an increase in the success rate.

For the custom architecture, the DP models with noise $\sigma = 1.3$ and clip norm $c = 1$ or $c = 3$ achieve higher or similar levels of robustness compared to the non-private baseline model. In contrast, models with higher clip norms can be attacked more successfully (see Figure 6.1a). The observation that some DP parameter combinations might be beneficial for robustness is in line with the findings by Tursynbek *et al.* [183] on their custom architecture. Yet, this behavior cannot be



(a) Custom architecture.



(b) LeNet architecture.

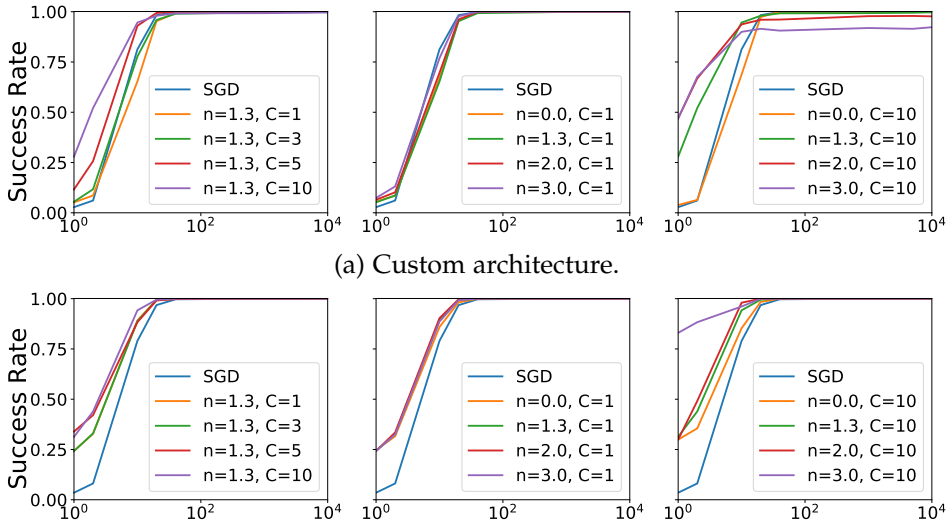
Figure 6.1.: **PGD$_\infty$ Success Rate vs. Adversarial Perturbation.** Results are plotted for 40 attack iterations [31].

observed for the LeNet model (see Figure 6.1b). For the LeNet model, the adversarial success rate when attacking the non-private baseline model was lower than the success rate observed for every DP parameter combination.

Similar to Tursynbek *et al.* [183], a plateau-like stagnation of the success rate for combinations with higher noise of $\sigma = 2$ or $\sigma = 3$ in combination with a high clip norm of $c = 10$ can be observed for the custom model. This might be interpreted as a robustness improvement using these settings. However, in the LeNet model, no similar behavior can be observed. Figure 6.1b (right) instead suggests that for $c = 10$, the higher the amount of noise, the lower the model robustness. Possible reasons for the difference between the two models might lie in their convergence. We observe that the custom model has larger gradient norms that the LeNet model, indicating worst convergence, which makes them more perceptible to input perturbations.

For both model architectures, the success rates of different noise values in combination with a small clip norm of $c = 1$ are very similar. While for the custom architecture, the attacks achieve a higher success rate for the non-DP model, for the LeNet models, the opposite applies.

Step-based Analysis. In the second experiment, the success rate of PGD$_\infty$ when attacking both architectures were evaluated depending on the number of iterations



(a) Custom architecture.

(b) LeNet architecture.

Figure 6.2.: **PGD_∞ Success Rate vs. Number of Attack Iterations.** Results are plotted for $\beta = 0.3$ [31].

or so-called *steps* for a fixed β of 0.3. The results suggest that, in general, the success rate increases with increasing numbers of iterations and finally reaches 100% (see Figure 6.2). However, in the custom model, with $\sigma = 2, \sigma = 3$, and $c = 10$, the success rate reaches a plateau after ~10 iterations and does not reach 100%. In contrast, for the LeNet architecture, no plateaus are reached. For the non-DP model highest robustness is reached with a sufficient amount of performed attack steps.

In summary, the experiments presented above confirm the findings by Tursynbek *et al*. [183]. Adversaries using PGD_∞ cannot successfully attack all their custom and privately trained model with a success rate of 100%. Based on this finding Tursynbek *et al*. concluded, that using the DPSGD optimizer during training simultaneously improves privacy, as well as robustness. Yet, the experiments presented in this section using the LeNet model already suggest, that this finding does not generalize to other architectures. Furthermore, in the next section, additional evidence will be presented showing that the evaluated private models do not show an increased level of robustness compared to their normally trained counterparts.

6.2.3 Robustness Evaluation with the BA_2 Attack

To further investigate the impact of DPSGD training on model robustness, the gradient-free BA_2 was used to attack the DP models. Table 6.2 depicts the results for adversarial perturbations of $\beta = 1$, and $\beta = 2$ against the custom and LeNet

Parameters	LeNet		Custom	
	$\beta = 1$	$\beta = 2$	$\beta = 1$	$\beta = 2$
SGD	27.5%	79.4%	19.1%	83.2%
$\sigma = 1.3, C = 1$	38.5%	75.7%	21.6%	70.9%
$\sigma = 1.3, C = 3$	41.3%	81.3%	26.6%	77.6%
$\sigma = 1.3, C = 5$	49.7%	82.1%	51.3%	94.2%
$\sigma = 1.3, C = 10$	57.2%	89.3%	86.9%	99.9%
$\sigma = 0, C = 1$	38.0%	71.3%	19.5%	65.2%
$\sigma = 1.3, C = 1$	38.5%	75.7%	21.6%	70.9%
$\sigma = 2, C = 1$	36.3%	76.7%	22.6%	72.4%
$\sigma = 3, C = 1$	40.9%	76.3%	26.0%	74.1%
$\sigma = 0, C = 10$	41.2%	77.9%	17.0%	71.7%
$\sigma = 1.3, C = 10$	57.2%	89.3%	86.9%	99.9%
$\sigma = 2, C = 10$	65.2%	94.7%	98.3%	100.0%
$\sigma = 3, C = 10$	91.5%	91.8%	93.6%	100.0%

Table 6.2.: **Success Rates of BA_2.** Displayed for different perturbation values β on both model architectures [31].

architectures. The attack was executed with 25,000 iterations. For more iterations, no increase in success rates could be observed.

For both model architectures, the results suggest that increasing the clip value or the amount of noise for a high clip value leads to increased adversarial vulnerability. The custom models with $\sigma = 2$ or $\sigma = 3$ and $c = 10$, that reached a plateau in the adversarial success rate for PGD_∞, are most vulnerable to BA_2 among all settings and for both model architectures. Solely for the parameter combinations of $c = 1$ or $\sigma = 0$ the BA_2 attack with perturbation $\beta = 2$ reaches a lower success rate than for the non-DP baselines. However, in general, the success rates of the DP models are higher than that of the non-DP ones.

In accordance with the findings of the previous section, the experiments here again show, that DP models are generally not more robust than their normally trained counterparts. The experiments even show that for certain parameter combinations the DP models are attacked even more easily using the BA_2 method.

6.2.4 Robustness Evaluation with CW_2 Attack

In the final experiment, the optimization-based CW_2 attack was used. This method generates adversarial examples in an unbounded manner. Hence, to determine the robustness towards CW_2, the adversarial perturbation magnitude β needed to achieve a 100% adversarial success rate is measured. Table 6.3 depicts the results of the experiments after 10,000 attack iterations.

Parameters	LeNet	Custom
SGD	$\beta = 1.21$	$\beta = 1.20$
$\sigma = 1.3, C = 1$	$\beta = 1.06$	$\beta = 1.37$
$\sigma = 1.3, C = 3$	$\beta = 1.03$	$\beta = 1.17$
$\sigma = 1.3, C = 5$	$\beta = 0.93$	$\beta = 0.80$
$\sigma = 1.3, C = 10$	$\beta = 0.83$	$\beta = 0.45$
$\sigma = 0, C = 1$	$\beta = 1.11$	$\beta = 1.41$
$\sigma = 1.3, C = 1$	$\beta = 1.06$	$\beta = 1.37$
$\sigma = 2, C = 1$	$\beta = 1.04$	$\beta = 1.31$
$\sigma = 3, C = 1$	$\beta = 1.01$	$\beta = 1.24$
$\sigma = 0, C = 10$	$\beta = 1.12$	$\beta = 1.38$
$\sigma = 1.3, C = 10$	$\beta = 0.83$	$\beta = 0.45$
$\sigma = 2, C = 10$	$\beta = 0.63$	$\beta = 0.29$
$\sigma = 3, C = 10$	$\beta = 0.15$	$\beta = 0.50$

Table 6.3.: **Success of CW_2.** Adversarial perturbation β required to achieve a 100% adversarial success rate within 10,000 iterations of CW_2 attack, rounded to two decimal points [31].

The values suggest that with increasing noise, or with increasing clip norm, the amount of perturbation needed to achieve a 100% success rate decreases. This indicates decreased adversarial robustness. For the LeNet architecture, the CW_2 attack requires smaller magnitude of perturbations on all DP models than on the non-private baseline models. The required perturbation budget decreases monotonically when increasing the clip value or noise. Interestingly, in the custom architecture with $c = 1$, or $\sigma = 0, c = 10$, a higher perturbation than in the non-private setting is required. Also, when setting the clip value to $c = 10$, and varying the amount of noise, $\sigma = 3$ requires higher perturbation than $\sigma = 2$, or $\sigma = 1.3$, hence, no monotonic decrease could be observed.

This experiment again underlines that DP models do not generally exhibit a higher level of robustness. First, the CW_2 attack is capable of generating adversarial examples with a success rate of 100%. Second, the required perturbation budget to generate the adversarial examples is in the majority of cases smaller for the DP models, than for the normally trained ones, indicating lower robustness.

6.3 Differential Privacy and Transferability

In addition to evaluating the success of directly attacking the models in a white-box setting, transferability attacks were conducted. Thereby, the possibility of transferring adversarial examples between DP models with different parameters, or between private and non-private models was quantified (see Figure 6.4).

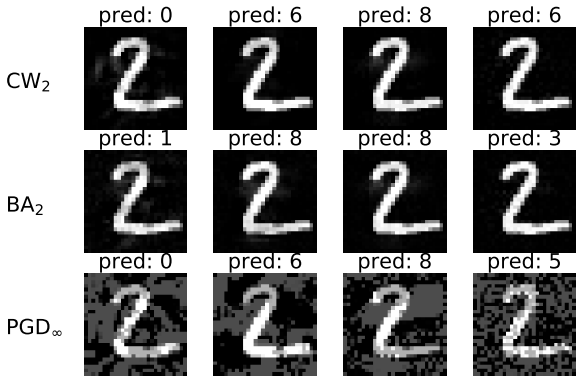


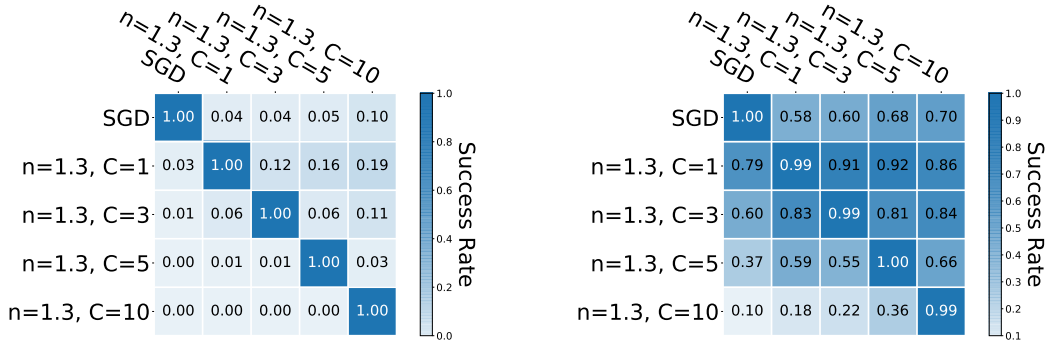
Figure 6.3.: **Adversarial Examples.** Generated with CW_2, BA_2, and PGD_∞ on the custom architecture models with different privacy settings (from left to right): SGD; $\sigma = 1.3, c = 1$; $\sigma = 3, c = 1$; $\sigma = 3, c = 10$ [31].

CW_2-Transferability In the first part of the experiment, the transferability of adversarial examples generated with the CW_2 attack was evaluated. For this purpose, for each of the models under attack, first, 1000 correctly classified test samples were chosen randomly. Then, the CW_2 attack was used to craft adversarial examples with 100% success rate on the respective surrogate models, *i.e.* models from which we want to transfer the adversarial examples to the original model under attack. Finally, the original model's accuracy on the generated adversarial examples was measured to determine the success rate of the attack. Figure 6.4a depicts the results for the custom model. Results for the LeNet models look similar.

The transferability of adversarial examples created with the CW_2 attack might be influenced by the different levels of applied perturbations: The perturbation budgets that lead to a 100% success rate in the CW_2 attack vary between models and tend to be lower for DP models (see Table 6.3).

PGD_∞-Transferability Therefore, in the second experiment, the transferability of adversarial examples generated with PGD_∞ and a constant perturbation of $\beta = 0.3$ was assessed. The procedure for determining the success rates was the same as for the CW_2 attack. Figure 6.4b summarizes the results of this test on the custom models. Again, results on LeNet models were similar.

The experiments suggest that for PGD_∞, the adversarial examples transfer significantly better than for CW_2. Still, the same trends can be observed for both scenarios: Adversarial examples seem to transfer less well from DP to non-DP models than the other way round. Additionally, adversarial examples generated on models with higher clip norms seem to transfer less well to other models than adversarial examples generated on models with lower clip norms. The adversarial examples generated on DP models with smaller clip norms ($c = 1$ and $c = 3$) transfer better to other DP models than the adversarial examples generated on the



(a) CW_2 attack. For perturbation values per model see Table 6.3.

(b) PGD_∞ attack. Perturbation $\beta = 0.3$ and 40 attack iterations.

Figure 6.4.: **Adversarial Transferability.** Transferability of generated adversarial examples between custom models. Adversarial examples were generated on models with settings depicted in the rows and evaluated against models with settings depicted in the columns. High success rates indicate that the adversarial examples transfer well, low success rates indicate low transferability [31].

non-private baseline models. Also, models trained with higher clip norms exhibit an increased vulnerability to transferability attacks compared to models with lower clip norms.

These results suggest that transfer attacks between models with different privacy settings are indeed successful. Again, the models that caused plateaus in the success rate when directly executing PGD_∞ against them seem to be the most vulnerable ones, according to this experiment. Another interesting observation is the fact that adversarial examples tend to transfer better between different DP models than from non-DP to DP models.

To investigate this effect further, adversarial examples created with the three considered attack methods on models with different privacy settings were visually examined (see Figure 6.3). The first column of the figures shows adversarial examples generated for the non-private baseline model. The results of the previous section and a visual inspection of the generated samples by CW_2 and BA_2 suggest that the more private the models are, the smaller the perturbation budget is required to fool the model with a 100% success rate.

The adversarial examples displayed for the PGD_∞ method all have the same perturbation budget of $\beta = 0.3$. Interestingly, the generated samples look significantly different between DP and non-DP models. Whereas in the non-DP model, the artifacts introduced into the images are grouped into regions, the higher the privacy gets, the more they resemble random noise and are not grouped into regions

anymore. The finding of this visual inspection is also reflected in the evaluation of the transferability experiments.

6.4 Discussion

This section discusses the findings of the experimental evaluation, their implications, possible underlying reasons, and provides an outlook on future research.

Influence of the DP Parameters. The experiments in this chapter have shown that, in general, DP models exhibit an increased adversarial vulnerability in comparison to non-DP models. In particular, the CW_2 and BA_2 attacks show clear trends: DP models with the same noise scale σ become less robust the higher the clip norm c is set. For relatively small clip norms, *e.g.* $c = 1$, an increase in the noise scale does not necessarily decrease model robustness, however, for larger clip norms, *e.g.* $c = 10$, this is the case. This suggests that in particular, the clip norm has a high influence on the resulting model's robustness. In particular, the aim of privacy protection and model robustness does not necessarily seem to align for all sets of parameters.

Gradient Masking. The principle of *gradient masking* [140] refers to methods that intentionally or unintentionally reduce the usefulness of an ML model's gradients for the generation of adversarial examples. As a consequence, gradient-based adversarial example generation performs less successfully. Since the models are often still vulnerable to non-gradient-based attacks, gradient masking results in a false sense of robustness.

When it comes to the success of adversarial examples, the following properties indicate sane and non-masked gradients [9, 41]. Violation(s) of these criteria can be an indicator for masked gradients.

1. Iterative attacks perform better than one-step attacks.
2. White-box attacks perform better than black-box attacks.
3. Gradient-based attacks perform better than gradient-free attacks.
4. Unbounded attacks should reach a 100% adversarial success rate.
5. Increasing the iterations within an attack should increase the adversarial success rate.
6. Increasing the distortion bound on the adversarial examples should increase the adversarial success rate.
7. White-box attacks perform better than transferability attacks using a similar substitute model.

Our experiments evaluate the adversarial success rate for the PGD_∞ with varying numbers of iterations (see Figure 6.2 on page 109) bringing the topic of gradient masking into focus. In the figure, we see that for some models with high clip norms and noise scales, increasing the number of iterations in the adversarial example

generation, at some point, does not increase the adversarial success rate anymore. This violates property 5. As a consequence, we observe that the adversarial success rate plateaus and never reaches 100%. A similar observation was reported by Tursynbek *et al*. [183]. The authors concluded from their findings that the respective models exhibit high robustness. However, we see that for the gradient-free BA_2 attack, the vulnerability of the plateauing models is much higher than that of the models that reach a 100% adversarial success rate. This rather indicates some form of gradient masking. As a consequence, we reason that the robustness estimation by prior work [183] might have been overly optimistic.

Differences between DP and non-DP Models. There are several possible explanations for the differences in robustness between DP and the baseline non-DP models. According to Demontis *et al*. [52], the larger the gradients in a target model, the larger the impact of attacks with adversarial examples. And indeed, we observe that the DP models' gradients are larger than normal models' gradients. This may be due to the use of DPSGD, in which Gaussian noise with a variance of $\sigma^2 c^2$ is added to the model gradients before the update. Hence, larger noise or clip norm parameters result in higher gradient magnitudes—potentially explaining the models' increased vulnerability. To counteract this factor, it might be helpful to regularize the gradients in DP model training.

Another factor for the increased vulnerability of DP models might be their decision boundaries. Tursynbek *et al*. [183] show that training with DPSGD affects the underlying geometry of the decision boundaries. In their example, it becomes visible that the DPSGD training results in more fragmented and smaller decision regions. This increases the chances of successfully generating an adversarial example with less perturbation.

In a similar vein, Demontis *et al*. [52] suggest that the loss surface of a model has an influence on the robustness against adversarial examples. They state that if the landscape of a model is very variable, it is much likely that slight changes to data points will encourage a change in the local optima of the corresponding optimization problem. As a consequence, the authors conclude that attack points might not transfer correctly to another model with a potentially more stable landscape. The experiments of this work depict that adversarial examples generated on DP models transfer less to normal models than the other way round and that adversarial examples crafted on models with higher clip norms transfer less than the ones from models with lower clip norms. This might also be due to the models' loss landscapes. Future work could, therefore, investigate the loss surface of the DP models more thoroughly.

Previous results by Papernot *et al*. [143] suggest that applying DP in combination with standard ReLU functions might lead to exploding activations in the resulting models. The authors suggest using sigmoid activation functions to counteract this

effect and, thereby, to improve the training process and achieve higher accuracy scores. In future work, it would be interesting to investigate whether this replacement of the activation functions might also be beneficial for the model's robustness. The authors also conclude that the models' activation functions have the largest influence on the success of DP training. However, the experiments in this work suggest, that the LeNet architecture might be more robust. Hence, when considering privacy in combination with security, the model architecture might be an important factor to consider as well.

Outlook on the Intersection between Privacy and Robustness. The results of this work raise the question of whether training ML models with DP does necessarily cause an increase in model vulnerability against adversarial examples. For the current state, the experiments suggest that achieving privacy can have a negative impact on model robustness. At the same time, this work also highlights a direction of research that might be worth pursuing in the future, namely controlling the gradients in DP model training. Pinot *et al*. [150] show that, in principle, Renyi-DP [126], which is used in the DPSGD, and adversarial robustness share equivalent goals. Therefore, future work could investigate how DP training can be adapted to simultaneously improve robustness and which factors, apart from the gradients, cause the current DP model's vulnerability.

6.5 Conclusion

Making NNs more private and more robust are important tasks that have long been considered separately. However, to solve both problems at the same time, it is beneficial to understand which impact they have on each other. This chapter addressed this question from a privacy perspective, evaluating how training ML models with DPSGD affects the robustness of the models. The experiments demonstrated that DPSGD training with certain parameters can cause a decrease in model robustness. By conducting a broad range of attacks against DP models with adversarial examples generated through different methods, we showed that the positive effects of DPSGD observed in previous work might be largely due to gradient masking, and therefore, provide a wrong sense of security. As a consequence, future work may further investigate the influence of DP training on the models. This could serve as a basis during parameter and architecture selection such that private training does not oppose the goal of security and can, hence, be applied also in critical scenarios.

Chapter 7

Conclusion

This chapter concludes our work. Therefore, it first presents a brief summary of the main results presented in previous chapters. Then, it highlights explicitly our contributions and the importance of this work. Moreover, it contains a suggestion of open problems around secure and private Machine Learning (ML) that future research could investigate. The chapter ends with a few final words on secure and private ML and an outlook on their importance in the broader scope of general trustworthy ML.

7.1 A Brief Summary

ML is applied in an ever increasing number of domains, dealing with critical tasks and highly sensitive data. This turns the ML models' security and the privacy of its training data into valuable targets for attackers. Therefore, it is of high importance to study these aspects as we did in this work.

In Chapter 2, we first introduced the concept of ML. Then, we described the protocol of Federated Learning (FL) in a formal language. Finally, we presented an overview of security and privacy threats against ML models relevant to the scope of this work. We also, on a high-level, introduced some defenses that can be applied to increase ML model security and privacy. More in depth, we formalized the concept of Differential Privacy (DP) and described its application in ML and FL. At the end of the chapter, we outlined the datasets used for experimental evaluations throughout this work.

Then, in Chapter 3, we presented a survey on the awareness in the field of secure and private ML that we conducted among ML practitioners to capture the current state of affairs. The results of this survey highlight a comparably low awareness among our participants, in particular when it comes to ML privacy. Additionally, the results show that ML practitioners face insecurities when implementing privacy regulations and, therefore, sometimes blindly trust in third-party services. These findings motivated our further research around privacy in ML, in particular with

a focus on raising awareness of existing privacy threats and studying the trust assumptions around third-party service providers. A better understanding of risks and trust assumptions in the field can inform the formulation of guidelines that then help ML practitioners in the implementation of security and privacy in ML.

Therefore, in Chapter 4 and Chapter 5, we studied privacy threats and trust assumptions in ML at the concrete example of FL. In Chapter 4, we first presented our observation on the inherent data leakage from the gradients of Neural Networks (NNs), even when these gradients are calculated over large mini-batches of data. Based on this observation, we developed a novel data extraction attack that relies on subtle manipulations of the model weights and architecture to increase the inherent data leakage from gradients. We applied this attack in the scenario of FL and showed that it enables the server to perform highly efficient data extraction attacks against the clients of the protocol. In Chapter 5, we then set out to study the trust assumptions in FL by analyzing the trust required in the central server to meet the theoretical privacy guarantees in practice. Our analysis showed that current FL protocols necessarily require trust in the server to provide practical privacy for the clients. It also suggested that the costs of implementing extensions for FL that provide privacy without trust in the server are, as of yet, prohibitive.

In Chapter 6, we investigated the impact of implementing theoretical privacy guarantees on the resulting ML models' security. Therefore, we trained NN models, using the Differentially Private Stochastic Gradient Descent (DPSGD) with varying parameters and measured the adversarial robustness of the models using different types of attacks. Our results highlight that some parameter combinations of the DPSGD training have a negative impact on model robustness. Additionally, the ML models' architecture and the resulting gradients play an important role in model robustness under privacy-preserving training.

7.2 Contributions

Secure and private ML is a highly relevant subfield of the general field of ML. Research around this topic, as well as this dissertation, aims at gaining a better understanding of the state of art and potential risks in order to design and extend existing defenses. In this work, we studied both the practical state of affairs as well as risks, trust assumptions, and potential defenses. Thereby, we made the following contributions.

The State of Secure and Private ML in Practice

First of all, we conducted a survey among ML practitioners to investigate their awareness and current practices in the field. Through our study, we distilled relevant research questions for this dissertation, such as studying existing tools and frameworks, which we did in Chapter 4 at the example of FL. In particular, we

aimed at raising awareness of privacy concerns and protection measures, the topics that ML practitioners in our survey were least familiar with. Our study also highlighted the trust that ML practitioners put into third-party services for implementing, in particular, ML privacy. This motivated our study on trust assumptions required to obtain privacy guarantees in FL that we presented in Chapter 5. Finally, through our study, we were able to provide valuable recommendations for future directions in making ML more secure and private by providing better guidance for ML practitioners.

Data Leakage from Model Gradients

In this work, we presented our novel observation that ML model gradients leak individual data points, even when the gradients are calculated over large mini-batches and on high dimensional data. We provided the theoretical background explaining this observation and performed an extensive experimental evaluation on the resulting practical data leakage.

Adversarial Weight Initialization and Data Extraction Attacks

Based on our observation on data leakage, we identified a novel attack vector against privacy in NNs, namely adversarial initializations of the model weights. We examined the potential of this attack vector by introducing different types of adversarial weights initializations. First and foremost, we introduced our adversarial weight initialization for fully-connected layers, which significantly increases the inherent data leakage from model gradients. Thereby, our adversarial weight initialization allows an attacker with access to the model gradients to perform highly efficient data extraction attacks. We showed how this attack vector can be used successfully in FL by the server to break the clients' privacy. Second, we proposed another adversarial weight initializations that cause unaltered forwarding of data points over fully-connected and convolutional model layers. They, thereby, make our data extraction attack applicable to a broader range of model architectures where they would otherwise not yield perfect data extractability. We also proposed measures for making such adversarial weight initializations for data forwarding more inconspicuous. Third, we sketched adversarial manipulations of model weights and architecture that enable higher-fidelity data extraction under the presence of privacy protection implemented by Distributed Differential Privacy (DDP) and Secure Aggregation (SA).

Studying Trust Assumptions in FL

We thoroughly studied the trust assumptions required in FL to achieve the theoretical privacy guarantees in practice. Therefore, we focused on the trust in the server and considered several extensions of the standard vanilla FL protocol which aim at providing increased levels of privacy. In particular, we were able to show that an

untrusted server can still disclose the clients' privacy, even when FL is protected by SA and DDP, currently considered the strongest deployment of the protocol. Based on our observations, we discussed protective measures that future work will have to provide to reduce the levels of trust required in the server. However, our investigation also suggested that, as of yet, the computational costs of such measures are prohibitive. As a consequence, we formulated recommendations for clients of FL protocols. These highlight how clients might be able to reduce data leakage, but also stress that they should not participate in FL at all if they mistrust the server orchestrating the protocol.

Impact of Differential Private Model Training on Robustness

Finally, we conducted a thorough study investigating the impact of training with DPSGD on the adversarial robustness of the resulting ML models. This complements prior work at the intersection of secure and private ML which primarily focused on the privacy implications of making ML models more robust. Our investigation showed that training with privacy guarantees can decrease the robustness of the resulting model. We believe that uncovering and understanding these kinds of trade-offs is of high importance. It will contribute to finding sweet spots between the different protection goals and also inform the design of ML methods that jointly optimize for both goals, security and privacy, at the same time.

7.3 Future Work

Based on the fruitful results of this work, we are able to derive promising future research directions. In the following, we enumerate them starting with directions that directly connect to our work and concluding with a broader outlook.

1. **Studying ML practitioners' workflows;**
   Such studies will help to gain a deeper understanding of the practitioners' coding practices and trust assumptions. This contributes in discovering what functionality, interfaces, and practices ML practitioners really rely on, which assumptions they hold, and which options they prefer. This can inform the (re-)design of libraries and tools to better implement security and privacy features for ML and to make the tools more usable. Thereby, in turn, general levels of ML security and privacy in practice can be elevated.

2. **Extending work on adversarial weight initializations;**
   In this work, we introduced adversarial weight initializations as a novel and versatile attack vector against privacy in NNs. Future work could build on our findings to refine this attack vector in order to extract more data, to reconstruct data with higher fidelity under the presence of noise, or to target specific data points or groups of data points with specific features.

3. **Extending adversarial weight initializations to black-box scenarios;**
   Our current adversarial weight initializations require an attacker with direct access to the model weights and the resulting gradients during training. It would be interesting for future work to investigate how the attack could be transformed into a black-box scenario. This could potentially be done by crafting specific training data points that poison the model weights in the same manner as an adversarial weight initialization. To reduce dependence on the model gradients for disclosing data privacy, the adversarial model weights could then encode private information directly into the model output.

4. **Improving privacy extensions for FL;**
   As shown in this work, current FL protocols require trust in the server to meet the theoretical privacy guarantees in practice. The computational costs of implementing existing privacy methods that forgo the need of trust seem to be prohibitive. Future work could, therefore, tend to develop privacy extensions for FL that require less computational and communication costs.

5. **Introducing model governance in FL;**
   ML Model governance [44] is concerned with the overall process to control access to the models, creating the right documentation, and monitoring models and their results. This allows to closely control model inputs and understand all the variables that might affect the corresponding outputs and thereby, of course, also privacy. However, without knowing which clients participate how often, and with what data in the protocol, it is hard to implement meaningful governance. In particular, it is difficult to implement some form of auditing, for example, in the sense of keeping track of the clients' final privacy guarantees. Future work could extend the concept of model governance into FL and develop meaningful methods to implement it.

6. **Extending research on trade-offs between privacy and robustness;**
   While our work shows that model training with DP has an impact on the model robustness, there is still room to explore the reasons behind this observation. Building on these insights, future work could develop methods that implement good trade-offs between both goals or jointly optimize for them by design.

7. **Considering security and privacy in the broader scope of trustworthy ML;**
   There exists a body of work that considers the intersection of different aspects that contribute to more trustworthy ML models. One example is the trade-off between privacy and fairness in ML, *e.g.* [177]. Future work could extend into considering not only trade-offs between two aspects, but multiple at the same time, such as the interplay between utility, privacy, fairness, and security. By jointly optimizing for several desirable properties at the same

time, this research would contribute to increasing the overall trustworthiness of ML models and applications relying on them.

7.4 Final Words

To conclude the entire work, we again want to stress the importance of securing ML models and protecting the privacy of their training data. Nowadays, ML is applied in a myriad of critical domains and the amount of sensitive data collected about each and every individual keeps on increasing. At the same time, the behavior of ML models and their decisions exhibit a large influence on our daily lives both as individuals and as a society. Therefore, the correct functioning of the ML models, even under concrete manipulations that target security and privacy needs to be ensured. But, as we mention during the outlook on future research directions, there are many more aspects of a desirable and trustworthy ML beyond security and privacy that need to be accounted for. In particular, the more diverse the tasks of our ML models become, the more the interplay between these different aspects gains importance. Therefore, we do not only need to implement ML security and privacy as isolated aspects, but we have to account for their impact on other aspects. This will help to implement ML such that it shapes our daily life and our society in a desirable way. We hope that our contributions within the field can raise attention among ML practitioners but also related researchers to take security and privacy into account when designing their ML systems.

Bibliography

[1] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and *et al.*. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. 2016.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 308–318.

[3] Martin Abadi, Ulfar Erlingsson, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Nicolas Papernot, Kunal Talwar, and Li Zhang. "On the protection of private information in machine learning systems: Two recent approches". In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE. 2017, pp. 1–6.

[4] Yasemin Acar, Christian Stransky, Dominik Wermke, Michelle Mazurek, and Sascha Fahl. "Security Developer Studies with GitHub Users: Exploring a Convenience Sample". In: *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. San Antonio, Texas: USENIX Association, 2017, pp. 81–95.

[5] Naman Agarwal, Peter Kairouz, and Ziyu Liu. "The skellam mechanism for differentially private federated learning". In: *Advances in Neural Information Processing Systems* 34 (2021).

[6] Adi Akavia, Max Leibovich, Yehezkel S Resheff, Roey Ron, Moni Shahar, and Margarita Vald. "Privacy-preserving decision trees training and prediction". In: *ACM Transactions on Privacy and Security* 25.3 (2022), pp. 1–30.

[7] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016).

[8] 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". In: *International Journal of Security and Networks* 10.3 (2015), pp. 137–150.

[9] 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*. PMLR. 2018, pp. 274–283.

[10] Office of the Australian Information Commissioner (OAIC). *Mobile Privacy: A Better Practice Guide for Mobile App Developers*. https://www.oaic.gov.au/privacy/guidance-and-advice/mobile-privacy-a-better-practice-guide-for-mobile-app-developers/, last accessed on: Jul. 2nd 2022. 2014.

[11] Rebecca Balebako and Lorrie Faith Cranor. "Improving App Privacy: Nudging App Developers to Protect User Privacy". In: *IEEE Security & Privacy* 12.4 (2014), pp. 55–58. DOI: 10.1109/MSP.2014.70.

[12] Rebecca Balebako, Abigail Marsh, Jialiu Lin, Jason I Hong, and Lorrie Faith Cranor. "The privacy and security behaviors of smartphone app developers". In: *Proceedings 2014 Workshop on Usable Security*. Citeseer, 2014. ISBN: 978-1-891562-37-2. DOI: 10.14722/usec.2014.23006.

[13] Borja Balle, Peter Kairouz, Brendan McMahan, Om Thakkar, and Abhradeep Guha Thakurta. "Privacy amplification via random check-ins". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4623–4634.

[14] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. "The security of machine learning". In: *Machine Learning* 81.2 (2010), pp. 121–148. ISSN: 1573-0565. DOI: 10.1007/s10994-010-5188-5.

[15] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. "Can machine learning be secure?" In: *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS*. New York, NY, USA: ACM, 2006, pp. 16–25. DOI: 10.1145/1128817.1128824.

[16] Maurice S Bartlett. "The effect of standardization on a χ 2 approximation in factor analysis". In: *Biometrika* 38.3/4 (1951), pp. 337–344.

[17] 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*. 2020, pp. 1253–1269.

[18] Nathan Benaich and Ian Hogarth. *State of AI Report 2019*. Technology. last accessed on: Jul. 2nd 2022. 2019. URL: https://www.stateof.ai/2019.

[19] Nathan Benaich and Ian Hogarth. *State of AI Report 2020*. last accessed on: Jul. 2nd 2022. 2020. URL: https://www.stateof.ai/2020.

[20] Yoav Benjamini and Yosef Hochberg. "Controlling the false discovery rate: a practical and powerful approach to multiple testing". In: *Journal of the Royal statistical society: series B (Methodological)* 57.1 (1995), pp. 289–300.

[21] Patrik Berander. "Using Students as Subjects in Requirements Prioritization". In: *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE '04.* New York, NY, USA: IEEE, 2004, pp. 167–176. DOI: 10.1109/ISESE.2004.1334904.

[22] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. "Evasion Attacks against Machine Learning at Test Time". In: *Advanced Information Systems Engineering*. Vol. 7908. Berlin, Heidelberg: Springer, 2013, pp. 387–402. DOI: 10.1007/978-3-642-40994-3_25.

[23] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. "Evasion attacks against machine learning at test time". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2013, pp. 387–402.

[24] Battista Biggio, Blaine Nelson, and Pavel Laskov. "Poisoning Attacks against Support Vector Machines". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. 2012, pp. 1467–1474.

[25] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. New York: Springer, 2006. ISBN: 9780387310732.

[26] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. "Prochlo: Strong privacy for analytics in the crowd". In: *Proceedings of the 26th Symposium on Operating Systems Principles*. 2017, pp. 441–459.

[27] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[28] Franziska Boenisch. "A Systematic Review on Model Watermarking for Neural Networks". In: *Frontiers in Big Data* 4 (2021).

[29] Franziska Boenisch, Verena Battis, Nicolas Buchmann, and Maija Poikela. ""I Never Thought About Securing My Machine Learning Systems": A Study of Security and Privacy Awareness of Machine Learning Practitioners". In: *Mensch und Computer 2021*. 2021, pp. 520–546.

[30] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. "When the Curious Abandon Honesty: Federated Learning Is Not Private". In: *arXiv preprint arXiv:2112.02918* (2021).

[31] Franziska Boenisch, Philip Sperl, and Konstantin Böttinger. "Gradient masking and the underestimated robustness threats of differential privacy in deep learning". In: *arXiv preprint arXiv:2105.07985* (2021).

[32] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečny, Stefano Mazzocchi, Brendan McMahan, et al. "Towards federated learning at scale: System design". In: *Proceedings of Machine Learning and Systems* 1 (2019), pp. 374–388.

[33] 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: *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1175–1191.

[34] Wieland Brendel, Jonas Rauber, and Matthias Bethge. "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models". In: *International Conference on Learning Representations*. 2018.

[35] BSI. *German IT Security Certificates*. https://www.bsi.bund.de/EN/Topics/Certification/certification_node.html, last accessed on: April 7th 2022. 2020.

[36] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. "Why and where: A characterization of data provenance". In: *International conference on database theory*. Springer. Berlin, Heidelberg: Springer, 2001, pp. 316–330.

[37] Lukas Burkhalter, Hidde Lycklama, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. "Rofl: Attestable robustness for secure federated learning". In: *arXiv preprint arXiv:2107.03311* (2021).

[38] Office of the Privacy Commissioner of Canada. *Seizing Opportunity: Good Privacy Practices for Developing Mobile Apps*. https://www.priv.gc.ca/en/privacy-topics/technology/mobile-and-digital-devices/mobile-apps/gd_app_201210/, last accessed on: Jul. 2nd 2022. 2012.

[39] Office of the Privacy Commissioner of Canada. *The Personal Information Protection and Electronic Documents Act (PIPEDA)*. https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/, last accessed on: Jul. 2nd 2022. 2019.

[40] Justin Cappos, Yanyan Zhuang, Daniela Oliveira, Marissa Rosenthal, and Kuo-Chuan Yeh. "Vulnerabilities as Blind Spots in Developer's Heuristic-Based Decision-Making Processes". In: *Proceedings of the 2014 Workshop on New Security Paradigms Workshop - NSPW '14*. The 2014 Workshop. Victoria, British Columbia, Canada: ACM Press, 2014, pp. 53–62. ISBN: 978-1-4503-3062-6. DOI: 10.1145/2683467.2683472.

[41] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. *On Evaluating Adversarial Robustness*. Living document; source available at https://github.com/evaluating-adversarial-robustness/adv-eval-paper/. 2019. URL: https://arxiv.org/pdf/1902.06705.

[42] Nicholas Carlini and David Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017 IEEE Symposium on Security and Privacy (SP) (San Jose, CA, USA). IEEE, 2017, pp. 39–57. ISBN: 978-1-5090-5533-3. DOI: 10.1109/SP.2017.49.

[43] Aaron Chalfin, Oren Danieli, Andrew Hillis, Zubin Jelveh, Michael Luca, Jens Ludwig, and Sendhil Mullainathan. "Productivity and selection of human capital with machine learning". In: *American Economic Review* 106.5 (2016), pp. 124–27.

[44] Varun Chandrasekaran, Hengrui Jia, Anvith Thudi, Adelin Travers, Mohammad Yaghini, and Nicolas Papernot. "SoK: Machine learning governance". In: *arXiv preprint arXiv:2109.10870* (2021).

[45] Wei-Ning Chen, Christopher A Choquette-Choo, and Peter Kairouz. "Communication Efficient Federated Learning with Secure Aggregation and Differential Privacy". In: *NeurIPS 2021 Workshop Privacy in Machine Learning*. 2021.

[46] Wei-Ning Chen, Christopher A Choquette-Choo, Peter Kairouz, and Ananda Theertha Suresh. "The Fundamental Price of Secure Aggregation in Differentially Private Federated Learning". In: *arXiv preprint arXiv:2203.03761* (2022).

[47] James Cheney, Laura Chiticariu, Wang-Chiew Tan, et al. "Provenance in databases: Why, how, and where". In: *Foundations and Trends® in Databases* 1.4 (2009), pp. 379–474.

[48] Christopher A Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. "CaPC learning: Confidential and Private Collaborative Learning". In: *arXiv preprint arXiv:2102.05188* (2021).

[49] Jacob Cohen. "A coefficient of agreement for nominal scales". In: *Educational and psychological measurement* 20.1 (1960), pp. 37–46.

[50] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified adversarial robustness via randomized smoothing". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1310–1320.

[51] Lee J Cronbach. "Coefficient alpha and the internal structure of tests". In: *psychometrika* 16.3 (1951), pp. 297–334.

[52] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. "Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks". In: *28th USENIX Security Symposium.* (Santa Clara, CA, USA). 2019, pp. 321–338. ISBN: 978-1-939133-06-9.

[53] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[54] Di Xie, Jiang Xiong, and Shiliang Pu. "All You Need is Beyond a Good Init: Exploring Better Solution for Training Extremely Deep Convolutional Neural Networks with Orthonormality and Modulation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.539.

[55] Thomas G Dietterich et al. "Ensemble learning". In: *The handbook of brain theory and neural networks* 2 (2002), pp. 110–125.

[56] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. *AdverTorch v0.1: An Adversarial Robustness Toolbox based on PyTorch*. https://github.com/BorealisAI/advertorch. 2019.

[57] Cynthia Dwork. "Differential Privacy". In: *Automata, languages and programming* (2006).

[58] Cynthia Dwork, Aaron Roth, et al. "The algorithmic foundations of differential privacy." In: *Foundations and Trends in Theoretical Computer Science* 9.3-4 (2014).

[59] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. "Amplification by shuffling: From local to central differential privacy via anonymity". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2019, pp. 2468–2479.

[60] Michael P. Fay and Michael A. Proschan. "Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules". In: *Statistics Surveys* 4.0 (2010), pp. 1–39. DOI: 10.1214/09-ss051.

[61] Ivan Flechais, Martina Angela Sasse, and Stephen Hailes. "Bringing security home: a process for developing secure and usable systems". In: *Proceedings of the New Security Paradigms Workshop*. Ed. by Christian Hempelmann and Victor Raskin. New York, NY, USA: ACM, 2003, pp. 49–57. DOI: 10.1145/986655.986664.

[62] Joseph L. Fleiss, Bruce Levin, and Myunghee Cho Paik. *Statistical Methods for Rates and Proportions*. Hoboken, New Jersey, USA: John Wiley & Sons, 2013.

[63] Liam H Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. "Robbing the Fed: Directly Obtaining Private Data in Federated Learning with Modified Models". In: *International Conference on Learning Representations*. 2021.

[64] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures". In: *CCS'15. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security : October 12-16, 2015, Denver, Colorado, USA* (New York, New York, USA). New York, New York: The Association for Computing Machinery, 2015. ISBN: 9781450338325. DOI: 10.1145/2810103.2813677.

[65] 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*. 2015, pp. 1322–1333.

[66] Galen Andrew, Steve Chien, and Nicolas Papernot. *Tensorflow/Privacy*. https://github.com/tensorflow/privacy, last accessed on: Feb. 2nd 2022. tensorflow, 2019. URL: https://github.com/tensorflow/privacy (visited on 02/12/2021).

[67] Stephen Gallant. "Perceptron-based learning algorithms". In: *IEEE Transactions on neural networks* 1.2 (1990), pp. 179–191.

[68] 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*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM, 2018, pp. 619–633. ISBN: 9781450356930. DOI: 10.1145/3243734.3243834.

[69] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. "Datasheets for datasets". In: *Communications of the ACM* 64.12 (2021), pp. 86–92.

[70] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. "Inverting gradients-how easy is it to break privacy in federated learning?" In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 16937–16947.

[71] Nastaran Gholizadeh and Petr Musilek. "Distributed Learning Applications in Power Systems: A Review of Methods, Gaps, and Challenges". In: *Energies* 14 (2021), p. 3654.

[72] Jairo Giraldo, Alvaro Cardenas, Murat Kantarcioglu, and Jonathan Katz. "Adversarial Classification Under Differential Privacy". In: *Proceedings 2020 Network and Distributed System Security Symposium* (Reston, VA). Reston, VA: Internet Society, 2020. ISBN: 1891562614. DOI: 10.14722/ndss.2020.23047.

[73] Antonious M Girgis, Deepesh Data, and Suhas Diggavi. "Differentially private federated learning with shuffling and client self-sampling". In: *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2021, pp. 338–343.

[74] Raja Giryes, Guillermo Sapiro, and Alex M. Bronstein. "Deep Neural Networks with Random Gaussian Weights: A Universal Classification Strategy?" en. In: *IEEE Transactions on Signal Processing* 64.13 (2016), pp. 3444–3457. ISSN: 1053-587X. DOI: 10.1109/TSP.2016.2546221.

[75] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". en. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (2011), pp. 315–323. ISSN: 1938-7228.

[76] Limesurvey GmbH. *LimeSurvey: An Open Source Survey Tool*. http://www.limesurvey.org, last accessed on: Feb. 2nd 2022. 2006.

[77] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets". In: *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Montreal, Canada, 2014.

[78] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[79] Thore Graepel, Kristin Lauter, and Michael Naehrig. "ML confidential: Machine learning on encrypted data". In: *International Conference on Information Security and Cryptology*. Springer. Berlin, Heidelberg: Springer, 2012, pp. 1–21.

[80] Matthew Green and Matthew Smith. "Developers are Not the Enemy!: The Need for Usable Security APIs". In: *IEEE Security & Privacy* 14.5 (2016), pp. 40–46. DOI: 10.1109/MSP.2016.111.

[81] Kathrin Grosse, Thomas A. Trost, Marius Mosbach, and Michael Backes. *Adversarial Initialization - when your network performs the way I want* -. 2019. URL: https://publications.cispa.saarland/id/eprint/2800.

[82] Xiaojie Guo, Zheli Liu, Jin Li, Jiqiang Gao, Boyu Hou, Changyu Dong, and Thar Baker. "VeriFL: Communication-efficient and fast verifiable aggregation for federated learning". In: *IEEE Transactions on Information Forensics and Security* 16 (2020), pp. 1736–1751.

[83] Philipp Hacker. "AI Regulation in Europe". In: *Available at SSRN 3556532* (2020).

[84] Saqib Hakak, Suprio Ray, Wazir Zada Khan, and Erik Scheme. "A Framework for Edge-Assisted Healthcare Data Analytics using Federated Learning". In: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020. DOI: 10.1109/bigdata50022.2020.9377873.

[85] Rob Hall and Stephen E Fienberg. "Privacy-preserving record linkage". In: *International conference on privacy in statistical databases*. Springer. 2010, pp. 269–283.

[86] Bartlomiej Hanus, John C Windsor, and Yu Wu. "Definition and multidimensionality of security awareness: Close encounters of the second order". In: *ACM SIGMIS Database: the DATABASE for Advances in Information Systems* 49.SI (2018), pp. 103–133.

[87] Jamie Hayes. "Provable trade-offs between private & robust machine learning". In: *arXiv preprint arXiv:2006.04622* (2020).

[88] Fengxiang He, Shaopeng Fu, Bohan Wang, and Dacheng Tao. "Robustness, privacy, and generalization of adversarial training". In: *arXiv preprint arXiv:2012.13573* (2020).

[89] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015. DOI: 10.1109/iccv.2015.123.

[90] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. "Deep models under the GAN: information leakage from collaborative deep learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017).

[91] Martin Höst, Björn Regnell, and Claes Wohlin. "Using Students as Subjects-A Comparative Study of Students and Professionals in Lead-Time Impact Assessment". In: *Empirical Software Engineering* 5.3 (2000), pp. 201–214.

[92] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. "Adversarial machine learning". In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence* (New York, New York, USA). ACM Press. New York, NY: ACM, 2011. ISBN: 9781450310031. DOI: 10.1145/2046684.2046692.

[93] Olakunle Ibitoye, M Omair Shafiq, and Ashraf Matrawy. "DiPSeN: Differentially Private Self-normalizing Neural Networks For Adversarial Robustness in Federated Learning". In: *arXiv preprint arXiv:2101.03218* (2021).

[94] Shubham Jain and Janne Lindqvist. "Should I Protect You? Understanding Developers' Behavior to Privacy-Preserving APIs". In: *Proceedings 2014 Workshop on Usable Security*. Workshop on Usable Security. San Diego, CA: Internet Society, 2014, p. 10. ISBN: 978-1-891562-37-2. DOI: 10.14722/usec.2014.23045.

[95] Hengrui Jia, Mohammad Yaghini, Christopher A. Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. *Proof-of-Learning: Definitions and Practice*. 2021. arXiv: 2103.05633 [cs.LG].

[96] Xin Jin and Jiawei Han. "K-Means Clustering". In: *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, pp. 563–564. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_425.

[97] Noah Johnson, Joseph P Near, and Dawn Song. "Towards practical differential privacy for SQL queries". In: *Proceedings of the VLDB Endowment* 11.5 (2018), pp. 526–539.

[98] Kaggle. *State of Machine Learning and Data Science 2020*. https://storage.googleapis.com/kaggle-media/surveys/Kaggle%20State%20of%20Machine%20Learning%20and%20Data%20Science%202020.pdf, last accessed on: April 7th 2022. 2020.

[99] Peter Kairouz, Ziyu Liu, and Thomas Steinke. "The distributed discrete gaussian mechanism for federated learning with secure aggregation". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5201–5212.

[100] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. "Practical and private (deep) learning without sampling or shuffling". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5213–5225.

[101] Henry F Kaiser. "A second generation little jiffy". In: *Psychometrika* 35.4 (1970), pp. 401–415.

[102] Henry F Kaiser and John Rice. "Little jiffy, mark IV". In: *Educational and psychological measurement* 34.1 (1974), pp. 111–117.

[103] Latif U. Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. "Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges". In: *IEEE Communications Surveys & Tutorials* 23.3 (2021), pp. 1759–1799. ISSN: 1553-877X. DOI: 10.1109/comst.2021.3090430.

[104] Trupti M Kodinariya and Prashant R Makwana. "Review on determining number of Cluster in K-Means Clustering". In: *International Journal* 1.6 (2013), pp. 90–95.

[105] Jakub Konečny, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. "Federated learning: Strategies for improving communication efficiency". In: *arXiv preprint arXiv:1610.05492* (2016).

[106] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[107] William H. Kruskal and W. Allen Wallis. "Use of Ranks in One-Criterion Variance Analysis". In: *Journal of the American Statistical Association* 47.260 (1952), pp. 583–621. ISSN: 01621459. URL: http://www.jstor.org/stable/2280779.

[108] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. "Adversarial machine learning-industry perspectives". In: *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE. New York, NY, USA: IEEE, 2020, pp. 69–75.

[109] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541.

[110] Yann LeCun, Corinna Cortes, and C. J. Burges. *MNIST handwritten digit database*. 2010.

[111] Yehida Lindell. "Secure multiparty computation for privacy preserving data mining". In: *Encyclopedia of Data Warehousing and Mining*. IGI global, 2005, pp. 1005–1009.

[112] Qiang Liu, Pan Li, Wentao Zhao, Wei Cai, Shui Yu, and Victor C. M. Leung. "A Survey on Security Threats and Defensive Techniques of Machine Learning: A Data Driven View". In: *IEEE Access* 6 (2018), pp. 12103–12117. DOI: 10.1109/ACCESS.2018.2805680.

[113] Ben Lorica Loukides Mike. *You Created a Machine Learning Application. Now Make Sure It's Secure.* https://www.oreilly.com/ideas/you-created-a-machine-learning-application-now-make-sure-its-secure, last accessed on: Jul. 2nd 2022. 2019.

[114] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[115] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083* (2017).

[116] Saeed Mahloujifar, Esha Ghosh, and Melissa Chase. "Property Inference from Poisoning". In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society. 2022, pp. 1569–1569.

[117] Mohammad Malekzadeh, Anastasia Borovykh, and Deniz Gündüz. "Honest-but-Curious Nets: Sensitive Attributes of Private Inputs Can Be Secretly Coded into the Classifiers' Outputs". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 825–844.

[118] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, and Zhifeng Chen *et al.*. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[119] Mary L McHugh. "The chi-square test of independence". In: *Biochemia medica* 23.2 (2013), pp. 143–149.

[120] 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* (2017), pp. 1273–1282. URL: http://proceedings.mlr.press/v54/mcmahan17a.html.

[121] Brendan McMahan and Abhradeep Thakurta. *Federated Learning with Formal Differential Privacy Guarantees*. https://ai.googleblog.com/2022/02/federated-learning-with-formal.html?m=1. Last Acccessed: March 25, 2022. 2022.

[122] H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. "A general approach to adding differential privacy to iterative training procedures". In: *arXiv preprint arXiv:1812.06210* (2018).

[123] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. "Learning Differentially Private Recurrent Language Models". In: *International Conference on Learning Representations*. 2018.

[124] Felipe A Mejia, Paul Gamble, Zigfried Hampel-Arias, Michael Lomnitz, Nina Lopatina, Lucas Tindall, and Maria Alejandra Barrios. "Robust or private? adversarial training makes models more vulnerable to privacy attacks". In: *arXiv preprint arXiv:1906.06449* (2019).

[125] 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)*. IEEE, 2019, pp. 691–706. ISBN: 978-1-5386-6660-9. DOI: 10.1109/SP.2019.00029.

[126] Ilya Mironov. "Rényi Differential Privacy". In: *IEEE 30th Computer Security Foundations Symposium - CSF 2017*. Ed. by IEEE Computer Security Foundations Symposium. IEEE Computer Security Foundations Symposium and CSF. Piscataway, NJ: IEEE, 2017. ISBN: 9781538632178. DOI: 10.1109/csf.2017.11.

[127] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.]: IEEE, 2016. ISBN: 9781467388511. DOI: 10.1109/cvpr.2016.282.

[128] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17*. Ed. by Bhavani Thuraisingham, Battista Biggio, David Mandell Freeman, Brad Miller, and Arunesh Sinha. New York, New York, USA: ACM Press, 2017, pp. 27–38. ISBN: 9781450352024. DOI: 10.1145/3128572.3140451.

[129] Kevin P. Murphy. *Machine learning: A probabilistic perspective*. Adaptive computation and machine learning series. Cambridge MA: MIT Press, 2012. ISBN: 9780262018029.

[130] Sarah Nadi, Stefan Krüger, Mira Mezini, and Eric Bodden. "Jumping through Hoops: Why Do Java Developers Struggle with Cryptography APIs?" In: *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*. The 38th International Conference. Austin, Texas: ACM Press, 2016, pp. 935–946. ISBN: 978-1-4503-3900-1. DOI: 10.1145/2884781.2884790.

[131] Yuki Nagai, Yusuke Uchida, Shigeyuki Sakazawa, and Shin'ichi Satoh. "Digital watermarking for deep neural networks". In: *International Journal of Multimedia Information Retrieval* 7.1 (2018), pp. 3–16.

[132] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. "Why Do Developers Get Password Storage Wrong?: A Qualitative Usability Study". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS '17*. The 2017 ACM SIGSAC Conference. Dallas, Texas, USA: ACM Press, 2017, pp. 311–328. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134082.

[133] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, and Matthew Smith. "Deception Task Design in Developer Password Studies: Exploring a Student Sample". In: *Fourteenth Symposium on Usable Privacy and Security, SOUPS 2018, Baltimore, MD, USA, August 12-14, 2018*. Ed. by Mary Ellen Zurko and Heather Richter Lipford. Berkeley, California, USA: USENIX Association, 2018, pp. 297–313.

[134] Dinh C. Nguyen, Ming Ding, Pubudu N. Pathirana, Aruna Seneviratne, Jun Li, and H. Vincent Poor. *Federated Learning for Internet of Things: A Comprehensive Survey*. Vol. 23. 2021. DOI: 10.1109/COMST.2021.3075439.

[135] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. "Adversarial Robustness Toolbox v1. 0.0". In: *arXiv preprint arXiv:1807.01069* (2018). https://github.com/Trusted-AI/adversarial-robustness-toolbox.

[136] Stanley RM Oliveira and Osmar R Zaiane. "Protecting sensitive knowledge by data sanitization". In: *Third IEEE International conference on data mining*. IEEE. New York, NY, USA: IEEE, 2003, pp. 613–616.

[137] Nicolas Papernot. "A Marauder's Map of Security and Privacy in Machine Learning: An overview of current and future research directions for making machine learning secure and private". In: *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security, CCS*. New York, NY, USA: ACM, 2018. DOI: 10.1145/3270101.3270102.

[138] Nicolas Papernot, Martin Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. *Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data*. Accepted to ICLR 17 as an oral. 2016. URL: https://arxiv.org/pdf/1610.05755.

[139] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, and Reuben Feinman *et al.*. *Technical Report on the CleverHans v2.1.0 Adversarial Examples Library*. https://github.com/cleverhans-lab/cleverhans. 2018. arXiv: arXiv:1610.007682.

[140] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. "Practical Black-Box Attacks against Machine Learning". In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (New York, New York, USA). Ed. by Ramesh Karri. [Place of publication not identified]: ACM, 2017. ISBN: 9781450349444. DOI: 10.1145/3052973.3053009.

[141] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. "Towards the science of security and privacy in machine learning". In: *arXiv preprint arXiv:1611.03814* (2016).

[142] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. "SoK: Security and Privacy in Machine Learning". In: *3rd IEEE European Symposium on Security and Privacy*. Los Alamitos, California: Conference Publishing Services, IEEE Computer Society, 2018. ISBN: 9781538642283. DOI: 10.1109/eurosp.2018.00035.

[143] Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. "Tempered Sigmoid Activations for Deep Learning with Differential Privacy". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10. 2021, pp. 9312–9321.

[144] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. "Eluding Secure Aggregation in Federated Learning via Model Inconsistency". In: *arXiv preprint arXiv:2111.07380* (2021).

[145] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, and Thirion *et al.*. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[146] Hai Phan, My T Thai, Han Hu, Ruoming Jin, Tong Sun, and Dejing Dou. "Scalable differential privacy with certified robustness in adversarial learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7683–7694.

[147] NhatHai Phan, Minh Vu, Yang Liu, Ruoming Jin, Dejing Dou, Xintao Wu, and My T Thai. "Heterogeneous Gaussian mechanism: Preserving differential privacy in deep learning with provable robustness". In: *arXiv preprint arXiv:1906.01444* (2019).

[148] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. "Privacy-Preserving Deep Learning: Revisited and Enhanced". en. In: International Conference on Applications and Techniques in Information Security. Springer, Singapore, 2017, pp. 100–110. DOI: 10.1007/978-981-10-5421-1_9.

[149] Olgierd Pieczul, Simon Foley, and Mary Ellen Zurko. "Developer-Centered Security and the Symmetry of Ignorance". In: *Proceedings of the 2017 New Security Paradigms Workshop on ZZZ - NSPW 2017*. The 2017 New Security Paradigms Workshop. Santa Cruz, CA, USA: ACM Press, 2017, pp. 46–56. ISBN: 978-1-4503-6384-6. DOI: 10.1145/3171533.3171539.

[150] Rafael Pinot, Florian Yger, Cédric Gouy-Pailler, and Jamal Atif. "A unified view on differential privacy and robustness to adversarial examples". In: *arXiv preprint arXiv:1906.07982* (2019).

[151] Rudiger Pryss, Manfred Reichert, Jochen Herrmann, Berthold Langguth, and Winfried Schlee. "Mobile Crowd Sensing in Clinical and Psychological Trials – A Case Study". In: *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*. IEEE, 2015. DOI: 10.1109/cbms.2015.26.

[152] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Françoise Beaufays. "Training production language models without memorizing user data". In: *arXiv preprint arXiv:2009.10031* (2020).

[153] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. "Foolbox Native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX". In: *Journal of Open Source Software*. Vol. 5, 53. https://github.com/bethgelab/foolbox. The Open Journal, 2020, p. 2607. DOI: 10.21105/joss.02607.

[154] Brandon Reagen, Woo-Seok Choi, Yeongil Ko, Vincent T Lee, Hsien-Hsin S Lee, Gu-Yeon Wei, and David Brooks. "Cheetah: Optimizing and accelerating homomorphic encryption for private inference". In: *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE. 2021, pp. 26–39.

[155] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. "Adversarial Attacks and Defenses in Deep Learning". In: *Engineering* 6.3 (2020). PII: S209580991930503X, pp. 346–360. ISSN: 2095-8099. DOI: 10.1016/j.eng.2019.12.012.

[156] Edo Roth, Daniel Noble, Brett Hemenway Falk, and Andreas Haeberlen. "Honeycrisp: large-scale differentially private aggregation without a trusted core". In: *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 2019, pp. 196–210.

[157] Edo Roth, Hengchu Zhang, Andreas Haeberlen, and Benjamin C Pierce. "Orchard: Differentially private analytics at scale". In: *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 2020, pp. 1065–1081.

[158] Mohammad Al-Rubaie and J Morris Chang. "Privacy-preserving machine learning: Threats and solutions". In: *IEEE Security & Privacy* 17.2 (2019), pp. 49–58.

[159] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. *A generic framework for privacy preserving deep learning*. https://github.com/OpenMined/PySyft. 2018. arXiv: 1811.04017 [cs.LG].

[160] Omer Sagi and Lior Rokach. "Ensemble learning: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018).

[161] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. "Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning". In: 29th USENIX Security Symposium (USENIX Security 20). 2020, pp. 1291–1308. ISBN: 978-1-939133-17-5.

[162] Iflaah Salman, Ayse Tosun Misirli, and Natalia Juristo. "Are Students Representatives of Professionals in Software Engineering Experiments?" In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. Vol. 1. New York, NY, USA: IEEE, 2015, pp. 666–676. DOI: 10.1109/ICSE.2015.82.

[163] Ravi S Sandhu and Pierangela Samarati. "Access control: principle and practice". In: *IEEE communications magazine* 32.9 (1994), pp. 40–48.

[164] Treasury Board Secretariat. "Directive on automated decision-making". In: *Ottawa (ON): Government of Canada (modified 2019-02-05* (2020).

[165] Educational Testing Service. *factor_analyzer: Open source Python module to perform exploratory and factor analysis*. https://factor-analyzer.readthedocs.io/en/latest/index.html/. 2019.

[166] Muhammad Shayan, Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. "Biscotti: A Blockchain System for Private and Secure Federated Learning". In: *IEEE Transactions on Parallel and Distributed Systems* 32.7 (2021), pp. 1513–1525. DOI: 10.1109/TPDS.2020.3044223.

[167] Yi Shi, Yalin E Sagduyu, Kemal Davaslioglu, and Jason H Li. "Active deep learning attacks under strict rate limitations for online API calls". In: *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE. New York, NY, USA: IEEE, 2018, pp. 1–6.

[168] 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 2017. 22-24 May 2017, San Jose, California, USA : proceedings*. Ed. by IEEE Symposium on Security and Privacy. IEEE Symposium on Security and Privacy et al. Piscataway, NJ: IEEE, 2017. ISBN: 9781509055333. DOI: 10.1109/sp.2017.41.

[169] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross J Anderson. "Manipulating sgd with data ordering attacks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18021–18032.

[170] Justin Smith, Brittany Johnson, Emerson Murphy-Hill, Bill Chu, and Heather Richter Lipford. "Questions Developers Ask While Diagnosing Potential Security Vulnerabilities with Static Analysis". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*. The 2015 10th Joint Meeting. Bergamo, Italy: ACM Press, 2015, pp. 248–259. ISBN: 978-1-4503-3675-8. DOI: 10.1145/2786805.2786812.

[171] Liwei Song, Reza Shokri, and Prateek Mittal. "Membership Inference Attacks Against Adversarially Robust Deep Learning Models". In: *2019 IEEE Security and Privacy Workshops (SPW)*. 2019 IEEE Security and Privacy Workshops (SPW) (San Francisco, CA, USA). IEEE, 2019, pp. 50–56. ISBN: 978-1-7281-3508-3.

[172] Mark Stamp. *Information security: principles and practice*. Hoboken, New Jersey, USA: John Wiley & Sons, 2011.

[173] State of California Department of Justice. *California Consumer Privacy Act (CCPA)*. https://oag.ca.gov/privacy/ccpa, last accessed on: Jul. 2nd 2022. 2018.

[174] Jeffrey Stylos and Brad A. Myers. "The implications of method placement on API learnability". In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2008, Atlanta, Georgia, USA, November 9-14, 2008*. Ed. by Mary Jean Harrold and Gail C. Murphy. New York, NY, USA: ACM, 2008, pp. 105–112. DOI: 10.1145/1453101.1453117. URL: https://doi.org/10.1145/1453101.1453117.

[175] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One Pixel Attack for Fooling Deep Neural Networks". In: *IEEE Transactions on Evolutionary Computation* 23.5 (2019), pp. 828–841. ISSN: 1089-778X. DOI: 10.1109/tevc.2019.2890858.

[176] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. "Provable Defense against Privacy Leakage in Federated Learning from Representation Perspective". In: *arXiv preprint arXiv:2012.06043* (2020).

[177] Vinith M Suriyakumar, Nicolas Papernot, Anna Goldenberg, and Marzyeh Ghassemi. "Chasing Your Long Tails: Differentially Private Prediction in Health Care Settings". In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. New York, NY, USA: ACM, 2021, pp. 723–734.

[178] Mikael Svahnberg, Aybüke Aurum, and Claes Wohlin. "Using students as subjects - an empirical evaluation". In: *Proceedings of the Second International Symposium on Empirical Software Engineering and Measurement, ESEM*. Ed. by H. Dieter Rombach, Sebastian G. Elbaum, and Jürgen Münch. New York, NY, USA: ACM, 2008, pp. 288–290. DOI: 10.1145/1414004.1414055.

[179] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[180] 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)*. 2016, pp. 601–618.

[181] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. "A hybrid approach to privacy-preserving federated learning". In: *Proceedings of the 12th ACM workshop on artificial intelligence and security*. 2019, pp. 1–11.

[182] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. "LDP-Fed: Federated learning with local differential privacy". In: *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. 2020, pp. 61–66.

[183] Nurislam Tursynbek, Aleksandr Petiushko, and Ivan Oseledets. "Robustness Threats of Differential Privacy". In: *arXiv preprint arXiv:2012.07828* (2020).

[184] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. "Embedding Watermarks into Deep Neural Networks". In: *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR*. Ed. by Bogdan Ionescu, Nicu Sebe, Jiashi Feng, Martha A. Larson, Rainer Lienhart, and Cees Snoek. New York, NY, USA: ACM, 2017, pp. 269–277. DOI: 10.1145/3078971.3078974.

[185] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[186] Dinusha Vatsalan, Peter Christen, and Vassilios S Verykios. "A taxonomy of privacy-preserving record linkage techniques". In: *Information Systems* 38.6 (2013), pp. 946–969.

[187] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, and SciPy 1.0 Contributors *et al.*. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[188] Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Berlin, Heidelberg: Springer, 2017.

[189] Christina Voskoglou. *What is the best programming language for Machine Learning?* https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7, last accessed on: Jul 2th 2022. 2017.

[190] Aidmar Wainakh, Ephraim Zimmer, Sandeep Subedi, Jens Keim, Tim Grube, Shankar Karuppayah, Alejandro Sanchez Guinea, and Max Mühlhäuser. "Federated Learning Attacks Revisited: A Critical Discussion of Gaps, Assumptions, and Evaluation Setups". In: *arXiv preprint arXiv:2111.03363* (2021).

[191] Jack Wallen. *Android 12 adds AI and machine learning with Private Compute Core but keeps your data secure*. https://www.techrepublic.com/article/android-12-adds-ai-and-machine-learning-with-private-compute-core-but-keeps-your-data-secure/. Last Acccessed: April 17, 2022. 2021.

[192] Yi Wang, Imane Lahmam Bennani, Xiufeng Liu, Mingyang Sun, and Yao Zhou. "Electricity Consumer Characteristics Identification: A Federated Learning Approach". In: *IEEE Transactions on Smart Grid* 12.4 (2021), pp. 3637–3647. ISSN: 1949-3053. DOI: 10.1109/tsg.2021.3066577.

[193] 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 2019 - IEEE Conference on Computer Communications*. IEEE, 2019. DOI: 10.1109/infocom.2019.8737416.

[194] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. "Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 501–518. ISBN: 978-1-939133-24-3.

[195] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. "Differentially private SQL with bounded user contribution". In: *arXiv preprint arXiv:1909.01917* (2019). https://github.com/google/differential-privacy.

[196] Glenn Wurster and P. C. van Oorschot. "The Developer Is the Enemy". In: *Proceedings of the 2008 Workshop on New Security Paradigms - NSPW '08*. The 2008 Workshop. Lake Tahoe, California, USA: ACM Press, 2008, p. 89. ISBN: 978-1-60558-341-9. DOI: 10.1145/1595676.1595691.

[197] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. "Verifynet: Secure and verifiable federated learning". In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 911–926.

[198] Haibo Yang, Xin Zhang, Prashant Khanduri, and Jia Liu. "Anarchic Federated Learning". In: *arXiv preprint arXiv:2108.09875* (2021).

[199] 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)*. 2018 IEEE 31st Computer Security Foundations Symposium (CSF). IEEE, 2018, pp. 268–282. ISBN: 978-1-5386-6680-7. DOI: 10.1109/CSF.2018.00027.

[200] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and and Pavlo Molchanov. *See through Gradients: Image Batch Recovery via GradInversion*. 2021.

[201] Hao Yu, Rong Jin, and Sen Yang. "On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7184–7193.

[202] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. "Salvaging federated learning by local adaptation". In: *arXiv preprint arXiv:2002.04758* (2020).

[203] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. "Protecting intellectual property of deep neural networks with watermarking". In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 2018, pp. 159–172.

[204] Shigeng Zhang, Shuxin Chen, Xuan Liu, Chengyao Hua, Weiping Wang, Kai Chen, Jian Zhang, and Jianxin Wang. "Detecting Adversarial Samples for Deep Learning Models: A Comparative Study". In: *IEEE Transactions on Network Science and Engineering* (2021), p. 1. ISSN: 2327-4697. DOI: 10.1109/tnse.2021.3057071.

[205] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. "iDLG: Improved deep leakage from gradients". In: *arXiv preprint arXiv:2001.02610* (2020).

[206] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu-an Tan. "Secure multi-party computation: theory, practice and applications". In: *Information Sciences* 476 (2019), pp. 357–372.

[207] Ligeng Zhu and Song Han. "Deep Leakage from Gradients". en. In: *Federated Learning*. Springer, Cham, 2020, pp. 17–31. DOI: 10.1007/978-3-030-63076-8_2.

Chapter A

Appendix

A.1 Zusammenfassung der Dissertation

In den letzten Jahren haben die Fortschritte im Bereich des Maschinellen Lernens (ML) dazu geführt, dass ML zunehmend in kritischen Anwendungen und auf hoch-sensiblen Daten eingesetzt wird. Dies rückt die Aspekte der Sicherheit und Privatheit in ML in den Fokus: ML-Modelle sollen korrekt funktionieren und nicht zu viele Informationen über ihre sensiblen Trainingsdaten preisgeben. Die Evaluierung und Umsetzung von ML-Sicherheit und Privatheit ist jedoch eine anspruchsvolle Aufgabe. Dies liegt in erster Linie daran, dass die Auswirkungen der derzeitigen ML-Praktiken auf ML-Sicherheit und Privatheit noch nicht vollständig verstanden sind. Folglich enthält die Palette der bekannten Risiken immer noch eine Vielzahl von Lücken. Ebenso bleiben die impliziten Annahmen unter denen ML-Sicherheit und Privatheit in einer bestimmten praktischen Anwendung erreicht werden können, oft unerforscht.

In dieser Arbeit stellen wir eine Studie über Sicherheit und Privatheit in ML vor, die dazu beiträgt, die bestehenden Limitierungen zu überwinden. Dafür geben wir zunächst einen Einblick in den aktuellen Stand der Sicherheit und Privatheit von ML in der Praxis, indem wir eine Umfrage unter ML-Entwicklern durchführen. Wir stellen fest, dass ML-Entwickler ein besonders geringes Bewusstsein für Privatheit in ML haben und bei deren Implementierung auf Dienste von Drittanbietern vertrauen. Diese Erkenntnis verdeutlicht die Notwendigkeit, das Thema Privatheit in ML eingehender zu untersuchen. Wir tun dies am Beispiel von Federated Learning (FL), da es sich bei FL um ein weit verbreitetes Protokoll handelt, das in Szenarien mit hunderttausenden Nutzern angewendet wird. In diesem Rahmen untersuchen wir Informationsleaks in ML-Modellen und zeigen, dass Modellgradienten direkt private Informationen über große Teile ihrer sensiblen Trainingsdaten preisgeben. Aufbauend auf diesen Erkenntnissen erweitern wir die bestehende Forschung zu Angriffen auf die Privatsphäre dieser Trainingsdaten, indem wir einen neuartigen Angriffsvektor vorschlagen: eine Manipulation der Initialisierung der Modellgewichte. Durch eine gründliche Untersuchung dieses Angriffsvektors, bewerten wir die Annahmen über das erforderliche Vertrauen, um sinnvolle Privatsphäregarantien in FL zu erhalten. Insbesondere konzentrieren wir uns auf Vertrauensannahmen bezüglich des Servers in FL. Um die Schnittstelle von ML-Sicherheit und Privatheit zu erkunden untersuchen wir schließlich, wie sich die Implementierung von Privatsphäregarantien auf die Robustheit der Modelle auswirkt. Zusammenfassend ist das Ziel dieser Arbeit Aufmerksamkeit auf die Bedeutung eines sicheren und Privatsphäre-freundlichen Designs von ML-Methoden lenken – insbesondere wenn diese Methoden in realen Anwendungsfällen eingesetzt werden.

A.2 Machine Learning Survey

This section provides additional material for Chapter 3.

A.2.1 Pilot Study

This section contains additional information on the participant demographics for the pilot study,the developer and student questionnaire for the pilot study, as well as the code books used to analyze free-text fields.

A.2.1.1 Participant Demography

Table A.1 and Table A.2 depict developer and student demographics, respectively.

Table A.1.: **Developer Demographics in Pilot Study.** Table depicts the country where the developers are currently working in, the time they have already been working in ML development, and their highest educational degree.

Austria	Estonia	France	Germany	Ireland	Italy	Poland	Slovenia	Spain	Switzerland	UK
2	2	3	25	1	2	2	1	1	1	1

1-3 years	4-6 years	7-9 years	10 years or more	Still planning career
22	10	3	4	2

High school or secondary school degree	Bachelor's degree	Master's degree or diploma	Doctorate
1	1	30	9

Table A.2.: **Student Demographics in Pilot Study.** Student demographics depicting the students' major, main area where they apply ML, and the time they have already been involved in ML grouped by their degrees.

	Bachelor degree	Master degree
Computer Science	11	18
Other	2	1
University	11	16
Job	1	2
Hobby	1	1
1 semester	8	6
1 semester-1 year	4	6
1-2 years	1	7

A.2.1.2 Questionnaire and Code Books

This section lists the developer and student questionnaire, as well as the code books used for free text analyses. Open Questions with free text fields are marked with an asterisk, question with single choice are indicated by the answer symbols ○ and question with multiple answer possibilities are represented by the answer symbols □. The [opt] questions only appeared optional based on the answer to the previous question.

Developer Questionnaire

1. *DEM1:* How long have you been working as a developer for machine learning (ML) applications?
   - ○ I have no experience in ML development.
   - ○ 1-3 years
   - ○ 4-6 years
   - ○ 7-9 years
   - ○ 10 years or more
   - ○ I am still planning my career as a ML developer.
   - ○ I have never worked as a ML developer
   - ○ ML development is a hobby
2. *DEM2:* Which country are you currently working in?
   - ○ Albania
   - ○ ...
   - ○ Zimbabwe
3. *DEM3:* What is the highest educational degree you have obtained?
   - ○ Less than high school or secondary school degree (i.e. Abitur, baccalauréat, A levels etc.)
   - ○ High school or secondary school degree
   - ○ Bachelor's degree
   - ○ Master's degree or diploma
   - ○ Doctorate
4. *IMP1:* How important or unimportant do you think it is to ensure the security of the machine learning models that you work with?
   - ○ Unimportant
   - ○ Of little importance
   - ○ Moderately important
   - ○ Important
   - ○ Very important
5. *AWA1:* In comparison to other machine learning developers, how aware would you say you are about possible attacks on machine learning systems? I believe, that my awareness is...
   - ○ Much lower than on average.
   - ○ A bit lower than on average.

- ○ Average.
- ○ A bit higher than on average.
- ○ Much higher than on average.

6. *AWA2:* How did you build your machine learning security awareness?
   - ☐ Through studies in university or other educational institution.
   - ☐ Through workshops and tutorials.
   - ☐ Learned through practice.
   - ☐ Through self-study.
7. *AWA3:* Are you, or is someone else in your working environment responsible of taking care of security of machine learning (ML) solutions?
   - ○ I am solely responsible for ML security.
   - ○ I am responsible for ML security, together with some others.
   - ○ Someone else is responsible for ML security.
   - ○ Nobody is responsible for ML security, but it is taken care of.
   - ○ Nobody is responsible for ML security, and it is not taken care of.
8. **AWA4:* What general risks in machine learning are you aware of? Please describe in a few words or sentences.
9. **AWA5:* For which possible security risks in machine learning systems have you ever implemented preventive solutions?
10. *IMP2:* How sensitive or non-sensitive is the most sensitive type of data that you use in your machine learning models?
    - ○ Very non-sensitive
    - ○ Somewhat non-sensitive
    - ○ Somewhat sensitive
    - ○ Very sensitive
11. *IMP3:* What kind of data do you deal with in your machine learning models? Please feel free to also leave a comment if you feel that further clarification is needed!
    - ☐ Images, video or audio
    - ☐ Text
    - ☐ Financial
    - ☐ Medical and health
    - ☐ Transportation and traffic
    - ☐ Customers and users
    - ☐ Weather and environment
    - ☐ Smart environment
    - ☐ Society
12. *IMP4:* Do your machine learning models deal with data of individiuals?
    I work with data that is...
    - ○ ...not related with humans.
    - ○ ...indirectly related with humans.
    - ○ ...directly related with humans.
13. *IMP5:* For what purpose(s) do you develop machine learning solutions? Please list your project types based on how often you engage in them. Please rank

the items from most often to least often, only including the ones you at least occasionally engage in. Your highest ranking item should be on the top right
- Industry
- Research
- Hobby

14. *SOL1:* Have you ever implemented solutions or taken other measures to prevent *poisoning attacks* in your ML models? Poisoning attack: an attacker is able to inject their own data records to your training data. Your model might thereby learn things that is not supposed to, as for example due to the shift of classification boundaries. This could in the prediction phase be exploited by the attacker to obtain certain labels for their input to the model.
    - ○ Yes, I have implemented solutions to prevent a poisoning attack.
    - ○ I am familiar with the poisoning attack, but have not implemented solutions to prevent it.
    - ○ No, I am not familiar with the poisoning attack.

*opt What kind of solutions have you implemented to prevent poisoning attacks? Please describe shortly the measures taken.

15. *SOL2:* Have you ever implemented solutions or taken other measures to prevent *evasion attacks* in your ML models? Evasion attack: an attacker modifies a data record in such a minimal way, that the record would probably still seem normal to a human observer. The completely from the one on the original input. We call such examples adversarial. An attacker could exploit this to fool your model to make a wrong prediction.modification however causes your ML model to make a prediction that differs An example of an evasion attack: a ML system was fooled to categorize a 3D-printed turtle as a gun by a particular pattern printed on it.
    - ○ Yes, I have implemented solutions to prevent an evasion attack.
    - ○ I am familiar with the evasion attack, but have not implemented solutions to prevent it.
    - ○ No, I am not familiar with the evasion attack.

*opt What kind of solutions have you implemented to prevent evasion attacks? Please describe shortly the measures taken.

16. *SOL3:* Have you ever implemented solutions or taken other measures to prevent *impersonation attacks* in your ML models? Impersonation attack: To impersonate an individual from your dataset, an attacker tries to imitate data records of their victim. They can use this to get unauthorized access, or to generate specific adversarial examples for that victim. Thereby, they could harm the victim with wrong predictions about them.
    - ○ Yes, I have implemented solutions to prevent an impersonation attack.
    - ○ I am familiar with the impersonation attack, but have not implemented solutions to prevent it.
    - ○ No, I am not familiar with the impersonation attack.

*opt What kind of solutions have you implemented to prevent impersonation attacks? Please describe shortly the measures taken.

17. *SOL4:* Have you ever implemented solutions or taken other measures to prevent *inversion attacks* in your ML models? Inversion attack: The aim of an inversion attack is to extract personal information from your ML model. An attacker could query your model to obtain knowledge about your underlying training data. They could use this knowledge to build their own model and harm you financially, or to breach the privacy of the users represented by your training dataset.
    - ○ Yes, I have implemented solutions to prevent an inversion attack.
    - ○ I am familiar with the inversion attack, but have not implemented solutions to prevent it.
    - ○ No, I am not familiar with the inversion attack.

*opt What kind of solutions have you implemented to prevent inversion attacks? Please describe shortly the measures taken.

opt Have you taken some other measures to prevent attacks against ML systems?
    - ○ Yes
    - ○ No

18. *PRA1a:* Have you implemented any solutions or taken other measures to prevent attacks against machine learning systems?
    - ○ Yes
    - ○ No
19. **PRA1b:* Please describe shortly the measures you have taken to prevent attacks against machine learning systems.
20. *PRAtableA:* Have you ever implemented a method for *data sanitization*? Data sanitizition: All your training data is cleaned from potentially malicious data points. Samples that have a negative impact on the model's prediction output might be discarded.
    - ○ Yes, I have implemented this method.
    - ○ I am familiar with this method, but have not implemented it.
    - ○ No, I am not familiar with this method.
21. *PRAtableB:* Have you ever implemented a method for *data provenance*? Data provenance: For all your training data, the provenance is clear and traceable. Your data pipeline and data storages are well documented and protected against intrusions.
    - ○ Yes, I have implemented this method.
    - ○ I am familiar with this method, but have not implemented it.
    - ○ No, I am not familiar with this method.
22. *PRAtableC:* Have you ever implemented a method for *adversarial training*? Adversarial training: Your model is trained partly on adversarial samples with corresponding labels to detect them as such and react adequately.
    - ○ Yes, I have implemented this method.
    - ○ I am familiar with this method, but have not implemented it.
    - ○ No, I am not familiar with this method.
23. *PRAtableD:* Have you ever implemented a method for *ensemble learning*? Ensemble learning: You are using several ML models as an ensemble for your

predictions. Hereby, different classifiers or different techniques for defence can be combined to mitigate adversarial samples.
   - ○ Yes, I have implemented this method.
   - ○ I am familiar with this method, but have not implemented it.
   - ○ No, I am not familiar with this method.

24. *PRAtableE:* Have you ever implemented a method for *observing model input at inference time*? Observing model input at inference time: You are observing the data that is presented to your model when it is deployed. ML models are most likely to fail when the data distribution at test time differs from the one at training time. By observing the input to your model, you can prevent an attacker using this fact to his advantage.
   - ○ Yes, I have implemented this method.
   - ○ I am familiar with this method, but have not implemented it.
   - ○ No, I am not familiar with this method.
25. *PRAtableE:* Have you ever implemented a method for *smoothing prediction output*? Smoothing prediction output: By changing the prediction output slightly before handing it to the user, or preventing sensitive outputs, you make it more difficult for an attacker to exploit your model. This is because it is more difficult for the attacker to reconstruct the model or to invert it.
   - ○ Yes, I have implemented this method.
   - ○ I am familiar with this method, but have not implemented it.
   - ○ No, I am not familiar with this method.
26. *PRAtable2A:* Have you ever implemented a method for *introducing delay for model interaction*? Introducing delay for model interacton: You do not allow unlimitedly many and unlimitedly frequent queries to your model by e.g. introducing a delay in your responses. Thereby, it gets more difficult for the attacker to build his own copy of your model that he can exploit or alter in order to harm you.
   - ○ Yes, I have implemented this method.
   - ○ I am familiar with this method, but have not implemented it.
   - ○ No, I am not familiar with this method.
27. *PRAtable2B:* Have you ever implemented a method for. *access control*? Access control: You ensure that each instance that interacts with your model only has the necessary access to perform its tasks. This can also include not giving the learned model access to the training data, once that training is completed.
   - ○ Yes, I have implemented this method.
   - ○ I am familiar with this method, but have not implemented it.
   - ○ No, I am not familiar with this method.
28. *PRAtable2C:* Have you ever implemented a method for *system security*? System security: You deploy your ML models on secure servers and protect (dedicated) hardware, like GPUs, TPUs etc. against attacks.
   - ○ Yes, I have implemented this method.
   - ○ I am familiar with this method, but have not implemented it.
   - ○ No, I am not familiar with this method.

29. *PRAtable2D:* Have you ever implemented a method for *differential privacy (DP)*? Differential privacy: DP gives strong mathematical guarantees on the privacy of the data that you are using. It assumes an attacker with maximal knowledge and provides an upper bound on possible privacy breaches. Privacy plays an important part in the security of your model against, e.g. inversion.
    - ○ Yes, I have implemented this method.
    - ○ I am familiar with this method, but have not implemented it.
    - ○ No, I am not familiar with this method.
30. *PRAtable2E:* Have you ever implemented a method for *homomorphic encryption*? Homomorphic encryption: Homomorphic encryption allows to perform operations on encrypted data without having to decrypt it. After all transformation, when data is decrypted, the result should be the same as if the operations were performed on the raw data.
    - ○ Yes, I have implemented this method.
    - ○ I am familiar with this method, but have not implemented it.
    - ○ No, I am not familiar with this method.
31. *PRAtable2F:* Have you ever implemented a method for *watermarking*? Watermarking: You poison the training data of your model yourself in order to have your model react to certain (secret) triggers. This can, amongst others, help to identify stolen copies of your model and protect your intellectual property.
    - ○ Yes, I have implemented this method.
    - ○ I am familiar with this method, but have not implemented it.
    - ○ No, I am not familiar with this method.
32. **PRAfinal:* Is there something else that you would like to mention about your work and security in machine learning?
33. *GDPR1:* How familiar are you with the requirements that the EU's General Data Privacy Regulation (GDPR) has on handling of personal data?
    - ○ Not at all familiar
    - ○ Slightly familiar
    - ○ Moderately familiar
    - ○ Very familiar
    - ○ Extremely familiar
34. *GDPR2:* Have you been dealing with fulfilling the requirements of GDPR in your machine learning development?
    - ○ Yes
    - ○ No
35. *GDPR3:* Have you experienced any issues with fulfilling the requirements of GDPR?
    - ○ Yes
    - ○ No

opt What kind of issues have you experienced with fulfilling GDPR requirements? Please describe in a few words or sentences.

36. *GDPR4:* To what extent has the GDPR caused a change in your machine learning security practices?

- ○ Not at all
- ○ Very little
- ○ Somewhat
- ○ To a great extent

37. **GDPR4a:* What kind of changes has the GDPR prompted in your machine learning security practices? Please describe in a few words or sentences.
38. *GDPR6:* To what extent has the GDPR caused a change in your security practices in general?
    - ○ Not at all
    - ○ Very little
    - ○ Somewhat
    - ○ To a great extent
39. **GDPR6a:* What kind of changes has the GDPR prompted in your security practices in general? Please describe in a few words or sentences.
40. **GDPR7:* Is there something else that you would like to mention about GDPR in the context of security of machine learning?

Student Questionnaire

1. *DEM4:* What degree are you currently studying?
    - ○ Bachelor degree
    - ○ Master degree
    - ○ PhD degree
    - ○ *Other:
2. *DEM5:* What program are you studying?
    - ○ Computer Science
    - ○ Mathematics
    - ○ Statistics
    - ○ *Other:
3. *DEM2s:* Which country are you currently studying in?
    - ○ Albania
    - ○ ...
    - ○ Zimbabwe
4. *DEM6:* For how long have you been involved with ML?
    - ○ This is my 1st semester
    - ○ 1 semester - 1 year
    - ○ 1 - 2 years
    - ○ Longer than 2 years
5. *DEM7:* Do you hope to work with ML after your graduation?
    - ○ Yes
    - ○ No
    - ○ Not sure
6. **AWA4:* What general risks in machine learning are you aware of? Please describe in a few words or sentences.

7. **AWA5:* For which possible security risks in machine learning systems have you ever implemented preventive solutions?
8. *IMP5s:* For what purpose(s) do you develop machine learning solutions? Please list your project types based on how often you engage in them. Please rank the items from most often to least often, only including the ones you at least occasionally engage in. Your highest ranking item should be on the top right.
    - University related projects and class work
    - Hobby
    - Job next to studies
9. *AWA6:* Have you heard about security risks for ML already?
    ○ Yes
    ○ No

Afterwards, question *SOL1* to *PRAfinal* have been the same as for the developers.

Code books

1. *AWA4*
    - Bad quality data, training bias
    - Interpretation bias, ethics
    - Data/Model security
    - Robustness
    - Attacks
        - Attacks on privacy
        - Attacks on model availability
        - Attacks of model integrity
    - Unexplainable models
    - Other
2. *AWA5*
    - Issues with data quality
    - Interpretation bias, ethics
    - Data/Model security
    - Issues with robustness
    - Leakage or attack on privacy
    - Issues with model availability
    - Issues with model integrity
    - Other
    - None
3. *GDPR467*
    - New policies or workflows
    - Security (e.g. access control)
    - Data anonymization and sanitization
    - Difficulties

- Training and awareness
- Critique and shortcomings
- Other

A.2.2 Full Study

This section contains additional material for the full study of the work described in Chapter 3.

A.2.2.1 Questionnaire

The final questionnaire of the full study is depicted in the following pages.

Security and Privacy in Machine Learning

This survey aims to analyze the state of the art in implementation of security measures to protect machine learning (ML) systems. We want to investigate to what extent machine learning practitioners are aware of the different types of attacks (from inside and outside) that their models are exposed to. We also want to learn which types of protective measures are used. Finally, we are interested in learning what kind of experiences developers have had with fulfilling the requirements of the European General Data Protection Regulation (GDPR).

We kindly ask all respondents to stick to the truth as best as they can and avoid exaggerating their statements in any direction, as this ensures the validity of the results. This survey will take approximately 10-15 minutes to fill out and consists of a maximum of 25 questions.

The survey is conducted by is conducted by the Fraunhofer Institute for Applied and Integrated Security (AISEC) in cooperation with the Fraunhofer Institute for Secure Information Technology (SIT) and the Freie Universität Berlin (FU).

- ❑ Indicate questions that allow for multiple answers.
- o Indicate questions that allow for exactly one answer.

Data security and consent note:

Participation in this study is voluntary. You can discontinue your participation at any time with no negative consequences, but information gathered from you up until the point of cessation of your participation may be used in the study. The data collected within this study include questionnaire items regarding your experience, awareness, and knowledge. These data can not be linked to your person. The research data are collected purely for scientific purposes. The research data are only available to the researchers of the research group. The research team deploys appropriate technical and organizational security precautions to protect personal data against disappearance, misuse, unlawful use, change, or destruction. The data collected of you within this survey are retained as long as is necessary for the purpose it should fulfil, or as long as the legislation requires. Any contact information we might collect of you is separated from other questionnaire data, including demographic information. Upon request you are provided with additional details of the general principles of this study and its progress, or of the results concerning yourself.

- o Agree
- o Disagree

[Only participants who gave their consent were forwarded to the questionnaire.]

Demographics

1 Which country are you currently working in?

- Afghanistan
  ...
- Zimbabwe

2 What is the highest educational degree you have obtained?

- Less than high school or secondary school degree (i.e. Abitur, baccalauréat, A levels etc.)
- High school or secondary school degree
- Bachelor's degree
- Master's degree or diploma
- Doctorate
- Other: [ ]

3 What is your current working situation?

- I am a student. (If you are also working in a machine learning related position at the same time, please check employee or self-employed, according to what applies to you!)
- I am an employee.
- I am self-employed.
- I am unemployed.

4 How long have you been working with machine learning (ML) - either professionally or as a hobby?

- I have never worked with ML
- 1-3 years
- 4-6 years
- 7-9 years
- 10 years or more

5 What are your daily machine learning (ML)-related tasks?

- ❑ Coordinating ML projects and workflows
- ❑ Applying ML libraries (tensorflow, scikit learn, ...)
- ❑ Developing custom ML applications (e.g. design custom neural networks for given tasks)
- ❑ Developing ML tools or libraries from scratch
- ❑ Data cleansing and preparation
- ❑ Data analysis
- ❑ Data collection
- ❑ Evaluation
- ❑ Deployment and maintenance
- ❑ Other: [ ]

6 For what field(s) do you apply machine learning?

- ❑ Industry
- ❑ Industrial research
- ❑ Academic research
- ❑ Hobby
- ❑ Other: [ ]

7 What is the size of the company you are working for?

** This question was not displayed for participants who had indicated being a student.*

- o Self-employed
- o 1-10 employees
- o 11-50 employees
- o 51-200 employees
- o 201-500 employees
- o 501-1000 employees
- o 1001-5000 employees
- o 5001-10,000 employees
- o 10,001+ employees

8 How are the product(s) that your division develops concerned with machine learning (ML)?

** This question was not displayed for participants who had indicated being a student.*

- o ML is key part of the product.
- o ML is included in the product but not key part.
- o ML is only used internally for marketing.
- o ML is only used internally for other purposes than marketing (e.g. to improve the product, finance, ...).

Data and Sensitivity

9 Is any of the data you work with sensitive?

Sensitive data means information that has to be be protected against unwarranted disclosure (e.g. private or confidential data).

- o No
- o Yes

10 Do your machine learning models deal with data of individiuals?

I work with data that is...

- ...not related to humans (any of my data).
- ...indirectly related to humans (at least some data that I work with).
- ...directly related to humans (at least some data that I work with).

11 What type of data do you deal with in your machine learning models?

- ❑ Images
- ❑ Video
- ❑ Audio/Sound
- ❑ Text
- ❑ Location data
- ❑ Metadata
- ❑ Sensor data
- ❑ Tabular data
- ❑ Other: [ ]

12 What domain does the data you are working with stem from?

- ❑ Financial
- ❑ Medical and health
- ❑ Transportation and traffic
- ❑ Customers and users
- ❑ Weather and environment
- ❑ Smart environment and IoT
- ❑ Social media
- ❑ Public security
- ❑ Other: [ ]

ML Security

13 In your opinion, how important or unimportant is it to ensure the security of your machine learning models?

- Unimportant
- Of little importance
- Moderately important
- Important
- Very important

14 How did you build your current knowledge about machine learning security?

- ❑ Through courses at university
- ❑ Through workshops and tutorials
- ❑ Through practice
- ❑ Through self-study (e.g. online tutorials, webinars, literature)

- ❑ Other: [ ]

15 Who takes care of the security of the machine learning (ML) models in your working environment? (If you are unemployed, please check the answer that applies to your previous job.)

** This question was not displayed for participants who had indicated being a student.*

- o I take care of my ML projects' security.
- o I solely take care of all ML security.
- o I take care of all ML security, together with some others.
- o A designated expert takes care of ML security.
- o Nobody takes care of ML security.
- o Make a comment on your choice here: [ ]

Attacks on ML

16 For the following attacks against machine learning (ML), please check what applies to you.

	Yes, I have implemented solutions to prevent this type of attack.	I am familiar with this type of attack, but have not implemented solutions against it, yet.	No, I am not familiar with this type of attack.
Inversion attacks Inversion attack: The aim of an inversion attack is to extract information from your ML model. An attacker could query your model to obtain knowledge about the underlying training data.	o	o	o
Impersonation attacks Impersonation attack: To impersonate an individual from your dataset, attackers try to imitate data records of their victims. They can use those records to get unauthorized access, or to develop specially tailored attacks against that victim.	o	o	o
Poisoning attacks Poisoning attack: During training, an attacker is able to inject their own data records into your training data. Your model might thereby learn things it is not supposed to, due to the shift of	o	o	o

classification boundaries. This could be exploited by the attacker in the prediction phase.			
Evasion attacks Evasion attack: At test time, an attacker modifies a data record in such a minimal way, that the record still seems normal to a human observer. The modification however causes your ML model to make a prediction that differs completely from the one on the original input. Adversarial examples are an instance of evasion attacks.	o	o	o

ML Security Practices

17 For the following libraries, related to private and secure machine learning (ML), please select what applies to you.

	I have already worked with this library.	I have heard about this library but have not used it, yet.	I have never heard about this library before.
Tensorflow Privacy	o	o	o
Cleverhans	o	o	o
PySyft	o	o	o
Google's Differential Privacy	o	o	o
Uber SQL Differential Privacy	o	o	o
AdverTorch	o	o	o
Foolbox	o	o	o
Adversarial Robustness Toolbox (ART)	o	o	o

18 Have you ever implemented a method for...

	Yes, I have implemented this method.	I am familiar with this method, but have not implemented it, yet.	No, I am not familiar with this method.
...data sanitization?	o	o	o

Data sanitizition: All your training data is cleaned from potentially malicious data points. Samples that have a negative impact on the model's prediction output might be discarded.			
...data provenance? Data provenance: For all your training data, the provenance is clear and traceable. Your data pipeline and data storages are well documented and protected against intrusions.	o	o	o
...adversarial training? Adversarial training: Your model is trained partly on adversarial samples with corresponding labels to detect them as such and react adequately.	o	o	o
...ensemble learning to make your ML models more secure? Ensemble learning: You group several ML models into an assembly for your predictions. Hereby, different classifiers or different techniques for defence can be combined to mitigate the success of attacks and to make the model more robust.	o	o	o
...observing model input at inference time? Observing model input at inference time: You are observing the data that is presented to your model when it is deployed. ML models are most likely to fail when the data distribution at test time differs from the one at training time. By observing the input to your model, you can prevent an attacker using this fact to his advantage.	o	o	o
...smoothing prediction output? Smoothing prediction output: By rounding or truncating the prediction output slightly, or preventing sensitive outputs, you make it more difficult for an attacker to reconstruct the model or to invert it.	o	o	o
...federated learning (FL)?	o	o	o

Federated Learning: FL is an ML technique in which the model is trained across multiple decentralized devices or parties on their local data samples. In contrast to traditional ML techniques, where all data samples are uploaded to one central server for training. In FL, no data samples are exchanged.			

19 Have you ever implemented a method for...

	Yes, I have implemented this method.	I am familiar with this method, but have not implemented it, yet.	No, I am not familiar with this method.
…introducing delay for model interaction? Introducing delay for model interacton: You do not allow unlimitedly many and unlimitedly frequent queries to your model by e.g. introducing a delay in your responses. Thereby, it gets more difficult for the attacker to build his own copy of your model that he can exploit or alter in order to harm you.	o	o	o
…access control to protect your ML models? Access control: You ensure that each instance that interacts with your model has only the access necessary to perform its tasks. This can also include not giving the learned model access to the training data, once that training is completed.	o	o	o
…system security to protect your ML models? System security: You deploy your ML models on secure servers and protect certain hardware components, such as GPUs, TPUs etc., against attacks.	o	o	o
...differential privacy (DP)? Differential privacy: DP gives strong mathematical guarantees on the privacy of the data that you are using. It assumes an attacker with maximal knowledge and provides an upper bound on possible privacy breaches.	o	o	o

...homomorphic encryption (HE)? Homomorphic encryption: HE allows to perform arithmetic operations directly on the encrypted data without having to convert it into plain text first. Each operation provides an encrypted result, which, when decrypted, corresponds to the result that would have been obtained if the operation had been performed on the unencrypted data.	o	o	o
...watermarking? Watermarking: You poison the training data of your model yourself in order to have your model react to certain (secret) triggers. This can, amongst others, help to identify stolen copies of your model and protect your intellectual property.	o	o	o
...privacy preserving record linkage? Privacy Preserving Record Linkage: Some attributes of your data, that individually do not seem too sensitive, can act as pseudo-identifier for a data point, when considered together. Privacy preserving record linkage transforms this weaknes into a strength, by calculating hash values over those values, so that data across multiple datasets can be shared by different parties without disclosing the sensitive attributes.	o	o	o

GDPR

20 How familiar are you with the requirements that the EU's General Data Privacy Regulation (GDPR) places on the handling of personal data?

- Not at all familiar
- Slightly familiar
- Moderately familiar
- Familiar
- Extremely familiar

21 In your work with machine learning, have you been dealing with the fulfilment of GDPR requirements?

- Yes
- I don't know
- No

22 To what extent has the GDPR caused a change in your machine learning security practices?

- Not at all
- Very little
- Somewhat
- Very much
- To a great extent

23 What kind of changes has the GDPR prompted in your machine learning security practises? Please describe in a few words or sentences.

Please write your answer here: [ ]

Final

24 If you would like to participate in a potential follow-up study, please enter your email address.

Please write your answer here: [ ]

25 If you would like to be informed about any publications resulting from this survey, please enter your email address.

Please write your answer here: [ ]

We highly appreciate the time you took to fill out this questionnaire. Your contribution supports the research community in advancing the field of security for machine learning!

Sincerely, your Fraunhofer AISEC, SIT and FU Berlin team.

For any further questions, please do not hesitate to contact us via: securemachinelearning[at]aisec.fraunhofer.de. Your answers have been transmitted, you can close this window now.

A.2.2.2 Participant Demography

The following Table A.3 and Table A.4 provide additional insights into the participants' demography, their working environment, and the data they are working with.

A.2.2.3 Hypotheses and Factor Analysis

The hypotheses we evaluated within our full study are grouped into two different families, represented in Table A.5 and Table A.6, respectively. Table A.7 presents the factors for the factor analysis used to estimate the latent construct awareness.

Table A.7.: **Items for Factor Analysis.** List of selected items for the factor analysis to be applied on. Only items with loadings > 0.45 were considered to correlate strong enough with the latent construct resulting in five variables being excluded from further analysis — namely *Adversarial Training*, *Observing model input at inference time*, *Smoothing prediction output*, *Federated Learning and System Security* — for which respective loadings are not reported.

Item	Factor 1
Inversion Attack	.669
Impersonation Attack	.588
Poisoning Attack	.663
Evasion Attack	.654
Data Sanitization	.623
Data Provenance	.543
Adversarial Training	
Ensemble Learning	.481
Observing model input at inference time	
Smoothing prediction output	
Federated Learning	
Introducing delay for model interaction	.543
Access Control	.532
System Security	
Differential Privacy	.548
Homomorphic Encryption	.539
Watermarking	.503
Privacy-preserving Record Linkage	.644
% of total variance	33.94
Cronbach's Alpha	0.86

Table A.3.: **Survey Participants' Demographics.** Summary of participants' background information. Demographics marked with an * allowed for multiple answers. Table taken from [29].

Area	
Europe	64
North America	9
South America	2
Asia	5
Australia	3
Education	
High school / Secondary school degree	2
Bachelor's degree	12
Master's degree	55
Doctorate	13
Other	1
Employment	
Employed	73
Self-employed	6
Unemployed	4
ML Application*	
Industry	38
Industrial Research	39
Academic Research	48
Hobby	17
Working experience in ML	
1-3 years	49
4-6 years	18
7-9 years	5
10 years or more	11
Company size (# of employees)	
Self-employed	6
1-10	5
11-50	8
51-200	10
201-500	12
501-1000	8
1001-5000	15
5001-10 000	7
More than 10 000	12

Table A.4.: **Summary of participants' ML working environment.** Demographics marked with an * allowed for multiple answers and, therefore, do not add up to the sample size of 83. Table taken from [29].

Domain(s) of the ML data*	
Customers and users	32
Smart environment and IoT	22
Medical and health	20
Transportation and traffic	19
Financial	19
Public security	11
Weather and environment	8
Social media	7
Other	22
Type(s) of data handled*	
Images	44
Sensor data	39
Tabular data	38
Text	37
Metadata	30
Location data	24
Video	17
Audio/Sound	15
Other	4
Daily ML Task(s)*	
Applying ML libraries	59
Data analysis	54
Evaluation	47
Data cleansing and preparation	45
Coordinating ML projects and workflows	44
Developing custom ML applications	36
Data collection	31
Deployment and maintenance	27
Developing ML tools / libraries from scratch	16
Role of ML in product development	
Key part of the product	47
Included in the product but not key part	28
Used internally for other than marketing	6
Used internally only for marketing	2

Table A.5.: **Hypotheses for RQ1.** Set of hypotheses to answer the research question RQ1. All hypotheses in this table form a hypothesis family and are considered as such in the process of correction for multiple comparisons. Table taken from [29].

No.	Hypotheses	Test statistic	p-value	p^*
H.1.1	H_0: Awareness does not correlate with the individual's perception of how important ML security is.	$r_{(81)} = .073$	.514	.62
H.1.2	H_0: Educational degree has no effect on the participants' level of awareness.	$\chi^2_{(4)} = .89$	.827	.827
H.1.3	H_0: Awareness does not correlate with years of working experience in ML.	$r_{(81)} = .36$	$< .001$	.005
H.1.4	H_0: Company size has no effect on the participants' level of awareness.	$\chi^2_{(8)} = 15.26$	.054	.109
H.1.5	H_0: Participants who build their own ML applications do not have a higher awareness than those who don't.	$U_{(41,42)} = 584.0$	.006	.018
H.1.6	H_0: There is no difference in the level of awareness between participants working with directly human-related data, indirectly or data that is not human-related at all.	$\chi^2_{(2)} = 4.09$	.129	.194

Table A.6.: **Hypotheses for RQ2.** Set of hypotheses to answer the research question RQ2. All hypotheses in this table form a hypothesis family and are considered as such in the process of correction for multiple comparisons. Table taken from [29]

No.	Hypotheses	Test statistic	p-value	p^*
H.2.1	H_0: Years of working experience do not correlate with the individual's perception of how important ML security is.	$r_{(81)} = .635$	.57	.685
H.2.2	H_0: Company size has no effect on who is responsible for securing ML models in the respective working environment.	$\chi^2_{(32)} = 43.11$	.091	.182
H.2.3	H_0: Educational degree has no effect on whether someone is responsible for implementing security solutions.	$U_{(47,35)} = 774.5$	.296	.395
H.2.4	H_0: The number of years of working experience with ML has no effect on whether someone is responsible for implementing security solutions.	$U_{(48,35)} = 728.5$	.124	.212
H.2.5	H_0: The introduction of the GDPR had no effect on the individuals' ML security practices - grouped by how human-related the data used is.	$\chi^2_{(2)} = 12.39$	.002	.005
H.2.5.1	H_0: The introduction of the GDPR had no effect on the individuals' ML security practices with respect to those working with non-human related data and those who work with indirectly related data.	$U_{(23,17)} = 97.0$	.002	.005
H.2.5.2	H_0: The introduction of the GDPR had no effect on the individuals' ML security practices with respect to those working with non-human related data and those who work with directly related data.	$U_{(23,23)} = 130.0$	$< .001$	.003
H.2.5.3	H_0: The introduction of the GDPR had no effect on the individuals' ML security practices with respect to those working with indirectly related data and those who work with directly related data.	$U_{(17,23)} = 171.5$	.254	.38
H.2.6	H_0: The four attacks described are all equally well known.	$\chi^2_{(3)} = 1.12$	.772	.772
H.2.7	H_0: The four attacks described are all equally often implemented.	$\chi^2_{(3)} = 1.31$	.727	.772
H.2.8	H_0: The 14 methods described are all equally well known.	$\chi^2_{(13)} = 93.19$	$< .001$	$< .001$
H.2.9	H_0: The 14 methods described are all equally often implemented.	$\chi^2_{(13)} = 209.0$	$< .001$	$< .001$

A.3 Privacy in Federated Learning

This section presents additional material for Chapter 4.

A.3.1 Visual Results for Data Extraction

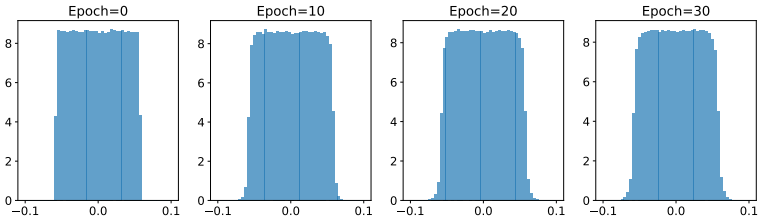
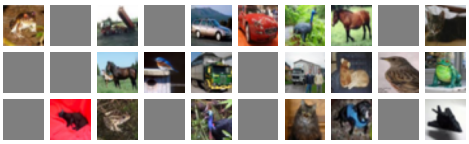


Figure A.1.: **Model Weight Distribution after Training.** Distribution of the first layer's weights of the FC-NN from Table 4.1 in Section 4.5 over training on the MNIST dataset. Weights at epoch zero were initialized with a random uniform distribution [30].

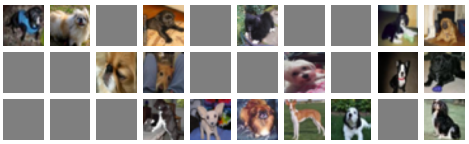
(a) Reconstructed data points. (b) Original data points.

Figure A.2.: **Extracted Data from MNIST.** Reconstruction success of our adversarial initialization: first 30 images from a mini-batch of 100 data points, extracted at the first fully-connected layer of the FC-NN from Table 4.1 in Section 4.5. Gray images indicate that the corresponding original data point could not be extracted individually from the model gradients [30].

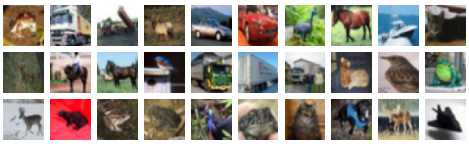
A.3.2 Data Extraction under Lossy Layers



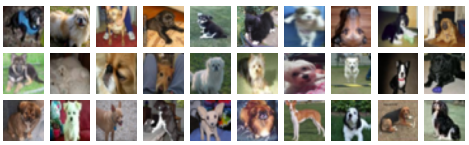
(a) Reconstructed data points from different classes.



(b) Reconstructed data points from class "dog".

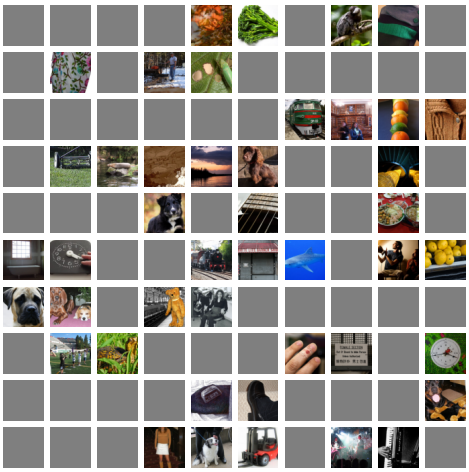


(c) Original data points from different classes.

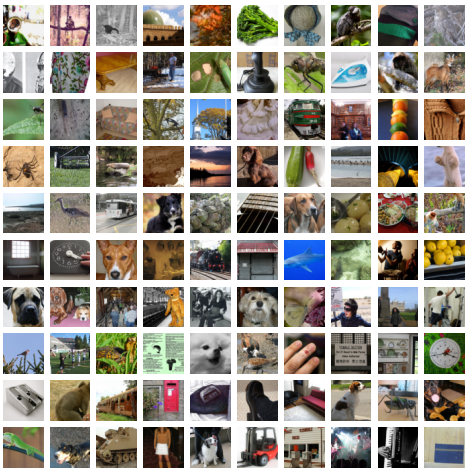


(d) Original data points from class "dog".

Figure A.3.: **Extracted Data from CIFAR-10.** Reconstruction success of our adversarial initialization: first 30 images from a mini-batch of 100 data points, extracted at the first fully-connected layer of the CNN from Table 4.1. Gray images indicate that the corresponding original data point could not be extracted individually from the model gradients [30].

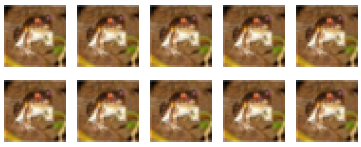


(a) Reconstructed data points.

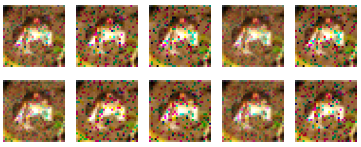


(b) Original data points.

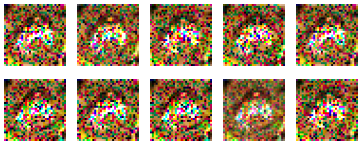
Figure A.4.: **Extracted Data from ImageNet.** Reconstruction success of our adversarial initialization: all reconstructed data points from a mini-batch of 100 data points, extracted at the first fully-connected layer of the CNN from Table 4.1. Gray images indicate that the corresponding original data point could not be extracted individually from the model gradients [30].



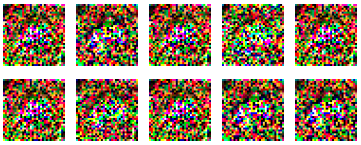
(a) Dropout with $p = 0.0$.



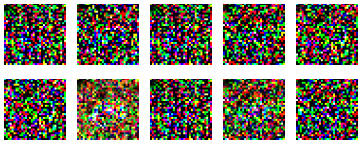
(b) Dropout with $p = 0.1$.



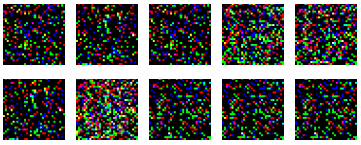
(c) Dropout with $p = 0.3$.



(d) Dropout with $p = 0.5$.



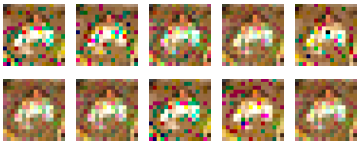
(e) Dropout with $p = 0.7$.



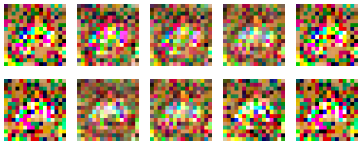
(f) Dropout with $p = 0.9$.

Figure A.5.: **Effect of Dropout for Mini-Batch Size 1.**

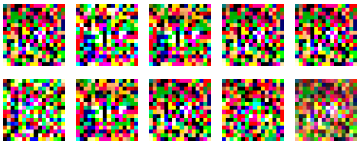
(a) Dropout with $p = 0.0$ and pooling.



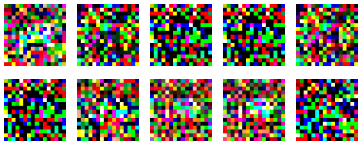
(b) Dropout with $p = 0.1$ and pooling.



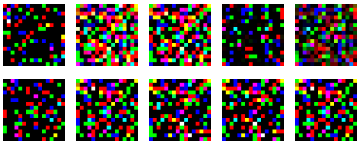
(c) Dropout with $p = 0.3$ and pooling.



(d) Dropout with $p = 0.5$ and pooling.

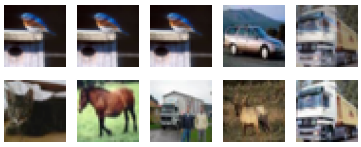


(e) Dropout with $p = 0.7$ and pooling.

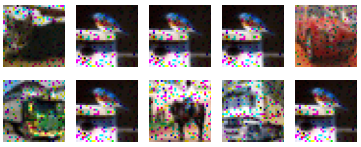


(f) Dropout with $p = 0.9$ and pooling.

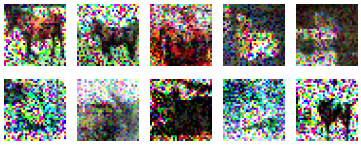
Figure A.6.: **Effect of Dropout+Pooling for Mini-Batch Size 1.**



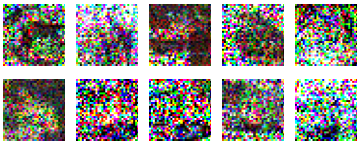
(a) Dropout with $p = 0.0$.



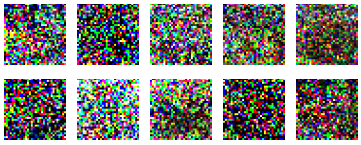
(b) Dropout with $p = 0.1$.



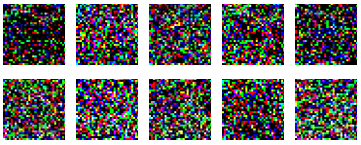
(c) Dropout with $p = 0.3$.



(d) Dropout with $p = 0.5$.



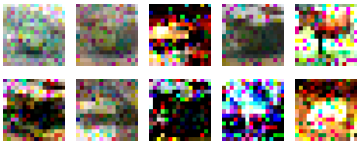
(e) Dropout with $p = 0.7$.



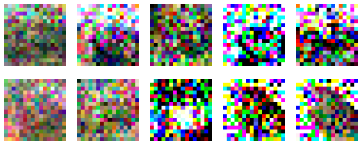
(f) Dropout with $p = 0.9$.

Figure A.7.: **Effect of Dropout for Mini-Batch Size 20.**

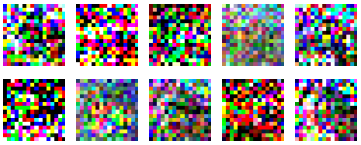
(a) Dropout with $p = 0.0$ and pooling.



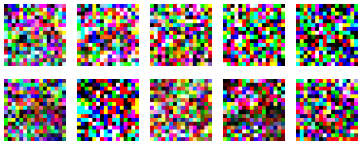
(b) Dropout with $p = 0.1$ and pooling.



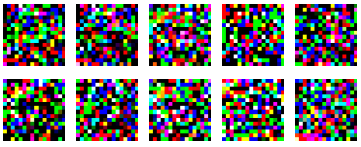
(c) Dropout with $p = 0.3$ and pooling.



(d) Dropout with $p = 0.5$ and pooling.



(e) Dropout with $p = 0.7$ and pooling.



(f) Dropout with $p = 0.9$ and pooling.

Figure A.8.: **Effect of Dropout+Pooling for Mini-Batch Size 20.**