Lessons Learned: Defending Against Property Inference Attacks

Joshua Stock[1][a], Jens Wettlaufer[2], Daniel Demmler[1][b] and Hannes Federrath[1]
[1] *Security in Distributed Systems, Universität Hamburg, Germany*
[2] *Institute of Electrical and Electronics Engineers (IEEE), U.S.A.*

Keywords: Machine Learning, Privacy Attacks, Property Inference, Defense Mechanisms, Adversarial Training.

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

1 INTRODUCTION

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

This work[1] focuses on ML with artificial neural networks (ANNs). After an ANN has been constructed, it can "learn" a specific task by processing big amounts of data in an initial training phase. During training, the connections between the network's nodes (or *neurons*) are modified such that the performance of the network regarding the specified task increases. After a successful training phase, the model, i.e., the network, is able to generalize, and thus enables precise predictions even for previously unseen data records. But while the model needs to extract meaningful properties from the training data to perform well in its dedicated task, it usually "remembers" more information than it needs to (Song et al., 2017). This can be particularly problematic if training data contains private and sensitive information such as intellectual property or health data. The unwanted manifestation of such information, coupled with the possibility to retrieve it, is called *privacy leakage*. In recent years, a new line of research has evolved around privacy leakage in ML models, which investigates privacy attacks and possible defense mechanisms (Rigaki and Garcia, 2020).

In this paper, we focus on a specific privacy attack on ML models: the *property inference attack* (PIA), sometimes also called *distribution inference* (Ateniese et al., 2015; Ganju et al., 2018). Given a trained ML model, PIAs aim at extracting statistical properties of its underlying training data set. The disclosure of such information may be unintended and thus dangerous as the following example scenarios show:

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

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

[a] https://orcid.org/0000-0003-3940-2229
[b] https://orcid.org/0000-0001-6334-6277
[1] This is an abbreviated conference version. For the full paper, please refer to (Stock et al., 2022).

Stock, J., Wettlaufer, J., Demmler, D. and Federrath, H.
Lessons Learned: Defending Against Property Inference Attacks.
DOI: 10.5220/0012049200003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 312-323
ISBN: 978-989-758-666-8; ISSN: 2184-7711

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

1.1 Contributions

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

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

Consequently, we have conducted further experiments with the explainable AI tool LIME (Ribeiro et al., 2016) and the visualization framework t-SNE (Van der Maaten and Hinton, 2008). Both provide evidence for the conjecture that properties are ubiquitous in the trained weights of an ANN, such that complete pruning of a property from a trained ANN is not possible without greatly limiting its utility.

In the full version of this paper, we additionally investigate the impact of simple training data preprocessing steps such as adding Gaussian noise to images of a training data set on the success rate of PIAs. This is meant as an inspiration for possible alternatives to techniques such as differential privacy, which has been established as a de-facto standard against many privacy attacks with the exception of PIAs (Rigaki and Garcia, 2020; Suri et al., 2022).

1.2 Organization of This Paper

Sect. 2 briefly explains ANNs, ML privacy attacks, our threat model and PIAs. Sect. 3 deals with an overview of related work. Our defense strategy property unlearning is presented in Sect. 4. Sect. 5 describes our property unlearning experiments, including our findings regarding its limitations. We further experimentally explore the reasons for these limitations via the explainable AI tool LIME and t-SNE visualization in Sect. 6. We summarize and discuss our findings in Sect. 7. Directions for future work are provided in Sect. 8, and Sect. 9 concludes this paper.

2 BACKGROUND

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

2.1 Artificial Neural Networks

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

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

2.2 Machine Learning Privacy Attacks

In general, privacy attacks against ML models extract information about training data of a target model $\mathcal{M}$ or the target model itself from its trained parameters. Some attacks, like membership inference (Shokri et al., 2017) extract information about a single record from a ML model. Other attacks try to recover the model itself (Papernot et al., 2017) or to recover the training data set or parts of it (Fredrikson et al., 2015). In contrast, this paper focuses on *property inference attacks* (PIAs), which reveal statistical properties of the entire training data set. This is not to be confused with *attribute inference attacks*, e.g., (Song and Shmatikov, 2020), which enable the adversarial recovery of sensitive attributes for *individual* data records from the training data set.

2.3 Threat Model

In the remainder of this paper, the following threat model is assumed: A model owner has trained and shared the model of an ANN. The owner wishes to keep their training data and its property $\mathbb{A}$ or $\mathbb{B}$ (a statistical property of the training data) secret. An example may be a company that has trained a model on its customer data and does not want to disclose any demographic information about their customers. If an attacker gets access to this model, they can perform a PIA and reconstruct the demographics of its training data, breaching the desired privacy. In another scenario, an attacker might want to gather information about a computer network before launching a malware attack. Such networks are often monitored by intrusion detection systems (IDS), which have been trained on network traffic to detect unusual behavior. Having access to this IDS model, the attacker could infer the OS most computers are running on in the system, or even detect specific vulnerabilities in the network, as demonstrated in (Ganju et al., 2018).

We assume that the attacker has full *white-box access* to the target model $\mathcal{M}$. This means that the attacker can access all parameters and some hyperparameters of $\mathcal{M}$: The adversary has a complete overview of the ANN architecture and can access the values of all weights and biases, as well as other useful hyperparameters of $\mathcal{M}$ such as the batch size during training, the learning rate and the number of training epochs. This helps the adversary to tailor their shadow models (see Sect. 2.4) as close to the target model as possible. In contrast, an adversary in a *black-box scenario* typically has oracle-access to the target model $\mathcal{M}$, allowing only to send queries to $\mathcal{M}$ and to analyze the corresponding results, i.e., the classification of a data instance.

As assumed in previous defenses against privacy attacks (Nasr et al., 2018; Song and Mittal, 2021; Tang et al., 2021), the attacker can access parts of the target model's training data, or knows a distribution of the training data, but cannot access the whole training data set. Information about the training data may also be reconstructed like in (Shokri et al., 2017), which is just as effective for privacy attacks (Liu et al., 2022).

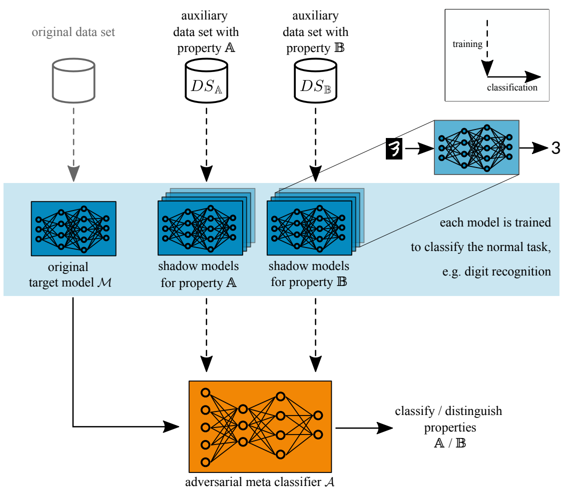

original data set
auxiliary data set with property $\mathbb{A}$
auxiliary data set with property $\mathbb{B}$
$DS_{\mathbb{A}}$
$DS_{\mathbb{B}}$
training
classification
3
original target model $\mathcal{M}$
shadow models for property $\mathbb{A}$
shadow models for property $\mathbb{B}$
each model is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

Figure 1: Property inference attack (PIA).

2.4 Property Inference Attacks (PIAs)

(Ateniese et al., 2015) were the first to introduce PIAs, with a focus on hidden Markov models and support vector machines. In this paper, we refer to the state-of-the-art PIA approach by (Ganju et al., 2018) who have adapted the attack to fully connected neural networks (FCNNs), a popular sub-type of ANNs. In a typical PIA scenario, an adversary has access to a trained ML model called *target model* $\mathcal{M}$, but not its training data. By using the model at inference time, a PIA enables the adversary to deduce information about the training data which the model has learned. Since the adversary's tool for the attack is a ML model itself, we call it *adversarial meta classifier* $\mathcal{A}$. Thus, the adversary attacks the target model $\mathcal{M}$ by utilizing the meta classifier $\mathcal{A}$ to extract a property from its training data.

A PIA typically involves the following steps (Ganju et al., 2018), see also Fig. 1:

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

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

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

4. After training the shadow models, use their resulting parameters (weights and biases) to train the adversarial meta classifier $\mathcal{A}$. During this training, the

meta classifier $\mathcal{A}$ learns to distinguish the parameters of target models that have been trained on data sets with property $\mathbb{A}$ and data sets with property $\mathbb{B}$, respectively. As a result, $\mathcal{A}$ is able to determine which of the properties $\mathbb{A}$ or $\mathbb{B}$ is more likely to be true for the training data of a given target model.

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

$\mathbb{A}$: proportion of male:female data instances 0.7:0.3

$\mathbb{B}$: male and female instances are equally present.

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

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

3 RELATED WORK

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

PIA Defense Strategies. Effective universal defense mechanisms against PIAs have not been discovered yet (Rigaki and Garcia, 2020). Differential privacy (Dwork et al., 2006) is a promising approach against other privacy attacks like membership inference (Rigaki and Garcia, 2020; Suri et al., 2022). However, it only slightly decreases the success rate of PIAs, since it merely limits the impact of each single input, but does not influence the presence of general properties in the training data set (Ateniese et al., 2015; Liu et al., 2022; Zhang et al., 2021).

(Ganju et al., 2018) propose *node multiplicative transformations* as another defense strategy. As long as an ANN uses ReLU or LeakyReLU as an activation function, it is possible to multiply the parameters of one layer by some constant and dividing the constants connecting it to the next layer by the same value without changing the result. Although they claim that this might be effective, this strategy is limited to ReLU and LeakyReLU activation functions and requires changes in the model architecture. In contrast, the approaches we test in this paper do not require any changes to the target model and do not require specific activation functions.

Other PIA Attacks. (Melis et al., 2019) explore PIAs in the context of collaborative learning: Herein, the adversary is a legitimate party in a collaborative setting, where participants jointly train a ML model via exchanging model updates – without sharing their local and private data. The authors present an *active* and a *passive* method to infer a property of the training data of another participant by analyzing the shared model updates of other participants.

Focusing on a black-box scenario, (Zhang et al., 2021) study both single- and multi-party PIAs for tabular, text and graph data sets. While their attack does not need access to the parameters of a target model, several hundreds of queries to the target model are needed for the attack to be successful.

An advanced PIA by (Mahloujifar et al., 2022) introduces *poisoning* as a way to ease the attack in a black-box scenario. This requires the adversary to control parts of the training data. In this adversarial training data set, the label of data points with a target property $\mathbb{A}$ are changed to an arbitrary label l. After training, the distribution of a target property can then be inferred by evaluating multiple queries to the target model – loosely summarized, the more often the label l is predicted, the larger the portion of samples with property $\mathbb{A}$ is in the training data set.

(Song and Shmatikov, 2020) propose a very similar attack to property inference, which we call attribute inference: They assume a ML target model which is partly evaluated *on-premise* and partly *in the cloud*. Their attribute inference attack reveals properties of a single data instance, e.g., whether a person wears glasses on a photo during the inference phase. In contrast, we focus on PIAs which reveal global properties about a whole training data set.

4 PROPERTY UNLEARNING

In this section we elaborate on our novel defense strategy against PIAs, which we call *property unlearning*. An overview of the approach is given in Figure 2.

As a prerequisite, an adversarial classifier $\mathcal{A}$ needs to be constructed. This is achieved as described in Sect. 2.4: constructing one auxiliary data set DS for each property $\mathbb{A}$ and $\mathbb{B}$, and training a set of *shadow models* for each property with the corresponding data sets $DS_{\mathbb{A}}$ and $DS_{\mathbb{B}}$. Note that when creating an adver-

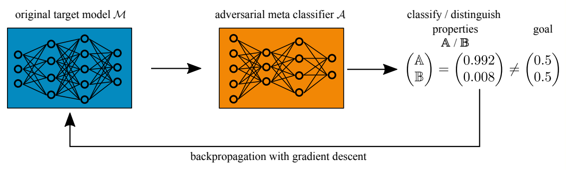

original target model $\mathcal{M}$
adversarial meta classifier $\mathcal{A}$
classify / distinguish properties $\mathbb{A}$ / $\mathbb{B}$
goal
$\begin{pmatrix}\mathbb{A}\\ \mathbb{B}\end{pmatrix} = \begin{pmatrix}0.992\\ 0.008\end{pmatrix} \neq \begin{pmatrix}0.5\\ 0.5\end{pmatrix}$
backpropagation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

Figure 2: Property unlearning as a defense strategy against PIAs.

sary as a preparation for protecting one's own model, the auxiliary data sets $DS_{\mathbb{A}}$ and $DS_{\mathbb{B}}$ can trivially be subsets of the original training data of the target model, since the model owner has access to the full training data set. This yields a strong adversarial accuracy as opposed to an outside adversary who might need to approximate or extract this training data first. The same holds for white-box access to the model, which is straightforward for the owner of a model. Hence, the training of a reasonably good adversarial meta classifier $\mathcal{A}$ ($> 99\%$ accuracy) as a first step of property unlearning is easily achievable for the model owner (see Sect. 5). As a second prerequisite, the target model $\mathcal{M}$, which the owner wants to protect, also needs to be fully trained with the original training data set – having either property $\mathbb{A}$ or $\mathbb{B}$.

To *unlearn* the property from $\mathcal{M}$, we use backpropagation. As in the regular training process, the parameters of the target model $\mathcal{M}$ are modified by calculating and applying gradients. But different from original training, property unlearning does not optimize $\mathcal{M}$ towards better classification accuracy. Instead, the goal is to disable the adversary $\mathcal{A}$ from extracting the property $\mathbb{A}$ or $\mathbb{B}$ from $\mathcal{M}$ while keeping its accuracy high.

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

Algorithm 1 shows pseudocode for the property unlearning algorithm. The termination condition for the while-loop in line 5 addresses the ability of the adversary $\mathcal{A}$: As long as $\mathcal{A}$ is significantly more confident for one of the properties, the algorithm needs to continue. After calculating the gradients g automatically via TensorFlow's backtracking algorithm in line 6, the actual unlearning happens in line 7. Here, the gradients are applied on the parameters of model $\mathcal{M}$, nudging them to be less property-

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

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

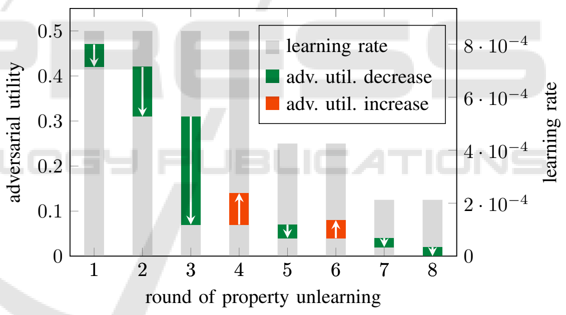


Figure 3: A visualized example of the decreasing adversarial utility during property unlearning with *one* adversary for a single target model $\mathcal{M}$. In each round, the adversarial utility of $\mathcal{M}$ either decreases further towards the goal of 0 (green bar), or the unlearning round is repeated with a smaller learning rate (after a red bar). The final result of round 8 is a completely *unlearned* target model $\mathcal{M}$ with an adversarial utility close to 0, see Algorithm 1.

revealing.

As described in Sect. 2.1, the learning rate controls how much the gradients influence a single step. If the parameters have been changed *too much*, the current $\mathcal{M}'$ gets discarded and the gradients are reapplied with half the learning rate (see line 12 and visualization in Fig. 3). Reducing the learning rate to its half has yielded the most promising results in our experiments.

The effect of property unlearning in between rounds of the algorithm is measured by the *adversar-*

ial utility, see lines 17–20. We calculate the adversarial utility by analyzing the adversary output Y. Recall that Y is a vector with k entries, with each entry Y_i representing the adversarially estimated probability that the underlying training data set of the target model $\mathcal{M}$ has property i. The adversarial utility is defined by the largest absolute difference of an entry Y_i to $\frac{1}{k}$ (see line 19). Remember that the goal of property unlearning is to nudge the parameters of $\mathcal{M}$ such that the output of the adversary is close to $\frac{1}{k}$ for all k entries in the output vector Y. The condition in line 9 therefore checks whether the last parameter update from $\mathcal{M}$ to $\mathcal{M}'$ was useful, i.e., whether the adversarial utility has decreased. Only if this is the case, the algorithm gets closer to the property unlearning goal. Otherwise, the last update in $\mathcal{M}'$ is discarded and the next attempt is launched with a lower learning rate. A visualization of an exemplary run is given in Fig. 3.

5 PROPERTY UNLEARNING EXPERIMENTS

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

Adversarial Property Inference Classifier. As described in Sect. 2.4, we use the attack approach by (Ganju et al., 2018). This means that each instance of an adversary $\mathcal{A}$ is an ANN itself, made up of multiple sub-networks ϕ and another sub-network ρ. Per data set, we train one such adversarial meta classifier $\mathcal{A}$, which is able to extract the respective properties $\mathbb{A}$ and $\mathbb{B}$ from a given target model.

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

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

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

This adversarial model $\mathcal{A}$ may then be employed in our property unlearning algorithm (see Sect. 4).

Data Sets and Network Architectures. We use three different data sets to evaluate our approach, as summarized in Table 1. For each data set and auxiliary data set DS_*, we train 2000 shadow models and 2000 target models. For faster training and a more realistic scenario, the auxiliary data sets DS_* are smaller. While the shadow models are used to train the adversaries $\mathcal{A}$, the target models $\mathcal{M}$ are the subjects of our experiments, i.e., we apply property unlearning on these target models and measure the resulting privacy-utility trade-off. The shadow models and target models share the same architecture per data set.

MNIST: is a popular database of labeled handwritten digit images. As in (Ganju et al., 2018), we distort all images with Gaussian noise (parameterized with mean = 35, sd = 10) in a copy of the database. We choose the property of having original pictures without noise ($\mathbb{A}_{MNIST}$) and pictures with noise ($\mathbb{B}_{MNIST}$). Our models for the MNIST classification task are ANNs with a preprocessing-layer to flatten the images, followed by a 128-neurons dense-layer and a 10-neuron dense-layer for the output.

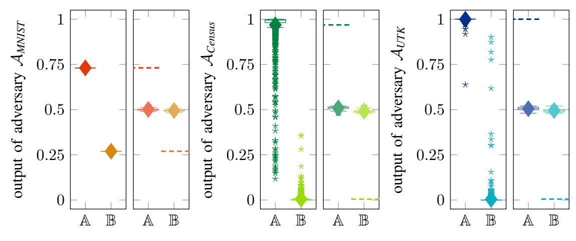
Census: is a tabular data set for income prediction. The property inference attack aims at extracting the ratio of male to female persons in the database, which is originally 2:1. The auxiliary data set for property $\mathbb{A}_{Census}$ $DS_{\mathbb{A}_{Census}}$ has a male:female ratio of 1:1, $DS_{\mathbb{B}_{Census}}$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

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

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

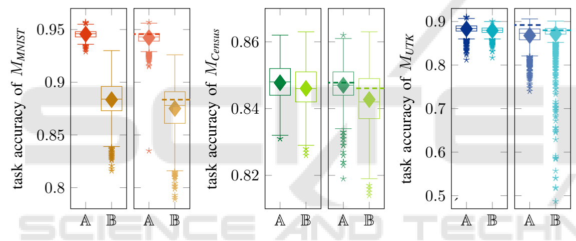
Table 1: The data sets used for the experiments. init.=initial, distrib.=distribution.

Experiment	Data set	Size	Target Property	$\lvert DS_{*}\rvert$	Shadow model accuracy	Init. PIA accuracy
$\mathcal{E}_{MNIST}$	(LeCun et al., 1998)	70K	Gaussian noise	12K	88.3–94.5%	100%
$\mathcal{E}_{Census}$	*Census Income Data Set*	48K	gender distrib.	15K	84.7%	99.3%
$\mathcal{E}_{UTK}$	(Zhang et al., 2017)	23K	race distrib.	10K	88.0–88.3%	99.8%



(a) $\mathcal{E}_{MNIST}$ (b) $\mathcal{E}_{Census}$ (c) $\mathcal{E}_{UTK}$

Figure 4: Each experiment before and after property unlearning, depicting the certainty of adversary $\mathcal{A}$ in classifying $\mathbb{A}$ and $\mathbb{B}$. The dashed lines represent the avg. accuracy *before* property unlearning was applied on 2000 target models.



(a) $\mathcal{E}_{MNIST}$ (b) $\mathcal{E}_{Census}$ (c) $\mathcal{E}_{UTK}$

Figure 5: Each experiment before and after property unlearning regarding the accuracy loss of the target models $\mathcal{M}$. The dashed lines represent the average accuracy values *before* property unlearning was applied on 2000 target models.

5.1 Experiment 1: Property unlearning

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

The figures in this section contain boxplot-graphs. Each boxplot consists of a box, which vertically spans the range between the first quartile Q_1 and the third quartile Q_3, i.e., the range between the median of the upper and lower half of the data set. The horizontal line in a box marks the median and the diamond marker indicates the average value.

MNIST. For the MNIST experiment $\mathcal{E}_{MNIST}$, the adversary classifies the properties $\mathbb{A}$ and $\mathbb{B}$ with high certainty in all instances before unlearning, see Fig. 4a. After unlearning, the adversary cannot infer the property of any of the MNIST target models $\mathcal{M}_{MNIST}$ – as intended. Meanwhile, the accuracy of the target models $\mathcal{M}_{MNIST}$ decreased slightly from an average of 94.6% by 0.4%P to 94.2% for models with property $\mathbb{A}$, respectively from 88.3% by 0.8%P to 87.5% for models with property $\mathbb{B}$ (see Fig. 5a). Recall that property $\mathbb{B}$ was introduced by applying noise to the training data, hence the affected models perform worse in general.

Census. Property unlearning was also successfully applied in the $\mathcal{E}_{Census}$ experiment to harden the target models $\mathcal{M}_{Census}$ against a PI adversary $\mathcal{A}_{Census}$, see Fig. 4b. Note that the performance of $\mathcal{A}_{Census}$ is not ideal for property $\mathbb{A}$, classifying some of the instances incorrectly. However, 99.3% of the 2000 instances were classified correctly by the adversary *before* property unlearning. As desired, the output of $\mathcal{A}_{Census}$ is centered around 0.5 for both properties after property unlearning. The magnitude of the target models' accuracy loss is small, with an average drop of 0.1%P for property $\mathbb{A}$ (84.8% to 84.7%) and 0.3%P (84.6% to 84.3%) for property $\mathbb{B}$, see Fig. 5b.

UTKFace. In the $\mathcal{E}_{UTK}$ experiment, property unlearning could be successfully applied to all models (see Fig. 4c) to harden the target models against PIAs. On average, the accuracy of the target models dropped by 1.3%P (from 88.2% to 86.9%) for models trained with the data set $DS_{\mathbb{A}}$ and by 0.1%P (from 87.9% to 87.8%) for target models trained with $DS_{\mathbb{B}}$, see Fig. 5c. This yields an average accuracy drop of 0.8%P across the target models for both properties (from 88.1% to 87.3%).

5.2 Experiment 2: Iterative Property Unlearning

In the previous section, the results of Experiment 1 have shown that property unlearning can harden a target model $\mathcal{M}$ against a single PI adversary, i.e., a specific adversarial meta classifier $\mathcal{A}$ (see Figure 2). The setup of Experiment 2 aims to improve that by generalizing the unlearning. Therefore, the same target model $\mathcal{M}$ is unlearned iteratively against

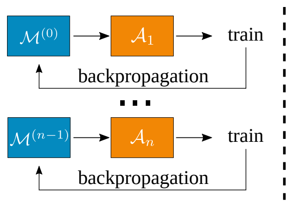


Figure 6: In reference to Fig. 2, iterative property unlearning works by performing single property unlearning for n different adversarial meta classifier instances $\mathcal{A}$ iteratively on a target model $\mathcal{M}^{(0)}$. The resulting target model $\mathcal{M}^{(n)}$ is then evaluated by additional m instances of $\mathcal{A}$.

a range of different adversary instances $\mathcal{A}$ (see Figure 6). The results of our experiments are based on 200 target models. We unlearn each initial target model $\mathcal{M}^{(0)}$ iteratively for n different adversarial meta classifiers $\mathcal{A}_i$, where $n = 15$. After that, the resulting iteratively unlearned target model $\mathcal{M}^{(n)}$ is tested by another distinct adversarial meta classifier. To increase the significance of our results, we choose to test the resulting target model $\mathcal{M}^{(n)}$ with $m = 5$ additional distinct adversarial meta classifiers. Furthermore, we apply a 4-fold cross validation technique to this constellation of in total 20 distinct adversarial classifiers. Finally, the results are plotted in boxplots similar to Experiment 1: Here, each boxplot is visualizing 200 (target models) $*$ 4 (folds) $*$ 5 (adversary outputs in a fold) $=$ 4000 data points.

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

The overall results on the MNIST data set in Figure 7 show the iterative unlearning process for property $\mathbb{A}$ and $\mathbb{B}$. Each column on the x-axis represents an iteration step of the iterative unlearning procedure. On the y-axis, the prediction of the adversary regarding the corresponding property is plotted, which is ideal for property $\mathbb{A}$ and $\mathbb{B}$ to be 0 and 1, respectively. The goal of property unlearning is $y = 0.5$, such that the attacker is not able to distinguish the property. Clearly, the second column shows that after applying property unlearning once, a distinct adversary, i.e., not the adversary which was involved in the unlearning process, is still able to infer the correct property for most target models $\mathcal{M}$. The plots show that after about ten iterations of property unlearning, the average output of the 5 testing adversaries converges towards an average of prediction probability 0.5 (for both properties $\mathbb{A}$ and $\mathbb{B}$). While this could be misinterpreted as ultimately reaching the goal of prop-

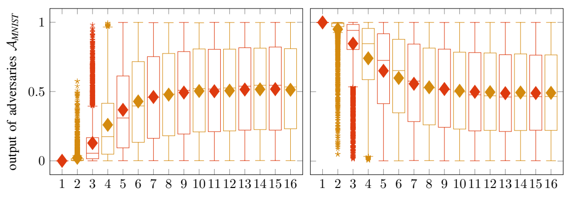


Figure 7: Results of iterative unlearning experiment for property $\mathbb{A}$ (left) and $\mathbb{B}$ (right). For each of the 200 target models $\mathcal{M}$, the predictions of all 5 testing adversaries are plotted along the y-axis before unlearning (first column) and after each unlearning iteration (other 15 columns).

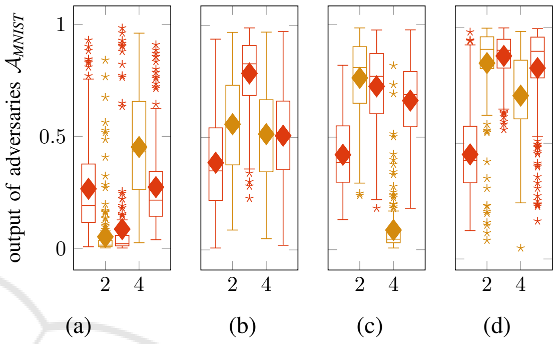


Figure 8: Individual adversary outputs after all 15 unlearning iterations for property $\mathbb{A}$ target models. Recall that before unlearning, all adversaries have correctly inferred property $\mathbb{A}$ by outputting $y = 0$.

erty unlearning, we introduce Figure 8 which paints a more fine-grained picture of the last column of Figure 7. Here, each of the four plots contain five independent boxplots corresponding to the five distinct test adversaries in one fold of the cross validation process. Each boxplot presents the prediction results of one adversary for the 200 independently unlearned target models $\mathcal{M}_i^{(15)}$ of the experiment.

While the plots of Figure 7 suggest that the adversaries' outputs are evenly spread across the interval $[0, 1]$ with both an average and median close to 0.5, Figure 8 shows that this is only true for the indistinct plot of all 4 experiments with 5 testing adversaries each. We want to point out three key observations:

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

2. For some adversary instances $\mathcal{A}$, target models have been "over-unlearned" by the 15 iterations with their output clearly nudged into opposite of their original output, e.g., adversary 3 in Fig. 8d.

3. Most importantly, other adversaries are still correctly inferring the property for most or even all 200 target models with high confidence after the 15 unlearning iterations, e.g., the second adversary in Fig. 8a.

5.3 Experiment Discussion

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

Experiment 1 (single property unlearning) shows that property unlearning is very reliable to harden target models against specific adversaries. However, Experiment 2 (iterative property unlearning) indicates that single property unlearning fails to generalize, i.e., protect against all PI adversaries of the same class. This is shown in Experiment 2 by putting each target model through 15 iterations of property unlearning with one distinct adversary per iteration. After this, some adversaries are still able to infer the original properties of all target models (see third key observation in Sect. 5.2). This means that in the worst case, i.e., for the strongest adversaries, 15 iterations of property unlearning do not suffice – while for other (potentially weaker) adversaries, 15 or even less iterations are enough to harden the models against them. In conclusion, property unlearning does not meet our goal of being a generic defense strategy, i.e., protecting against a whole class of adversaries instead of a specific adversary.

6 EXPLAINING PI ATTACKS

To explore the reasons behind this limitation of property unlearning, we use the explainable AI tool by (Ribeiro et al., 2016): **LIME** (*Local Interpretable Model-agnostic Explanations*) allows to analyze decisions of a black-box classifier by permuting the values of its input features. By observing their impact on the classifier's output, LIME generates a comprehensible ranking of the input features.

Recall that in the previous experiment (Sect. 5.2), we have seen that adapting the weights of a target model $\mathcal{M}$ s.t. an adversarial meta-classifier $\mathcal{A}_1$ cannot launch a successful PIA does not defend against another adversarial meta-classifier $\mathcal{A}_2$ trained for the same attack. Therefore, we use LIME to see whether different meta-classifiers $\mathcal{A}_1$ and $\mathcal{A}_2$ rely on the same weights of a target model $\mathcal{M}$ to infer $\mathbb{A}$ or $\mathbb{B}$.

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

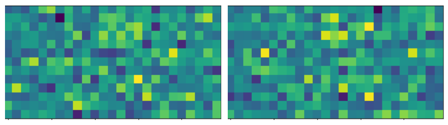


Figure 9: LIME produced, partial heat maps of different meta-classifier instances $\mathcal{A}_1$ and $\mathcal{A}_2$ for the same MNIST target model $\mathcal{M}$. Dark pixels represent parameters with high impact on the decision of $\mathcal{A}$, yellow pixels imply a low impact.

LIME Results. We have instantiated LIME with two property inference meta-classifiers $\mathcal{A}_1$ and $\mathcal{A}_2$ to explain their output for the same MNIST target model instance $\mathcal{M}$. The output of LIME is a heat map representing the weights and biases of $\mathcal{M}$, see Fig. 9. For practical reasons, we have only visualized the first 784 pixels of the heat map and transformed them to a two-dimensional space. Although $\mathcal{A}_1$ and $\mathcal{A}_2$ are trained in the same way and with the same shadow models (see Sect. 5), the two heat maps for classifying the property of the same target model $\mathcal{M}$ in Fig. 9 are clearly different: While some of $\mathcal{M}$'s weights have similar importance, i.e., the heat map pixels have a similar color, many weights have very different importance for the two adversarial meta classifiers $\mathcal{A}_1$ and $\mathcal{A}_2$.

To understand why meta-classifiers can rely on different parts of target model parameters to infer a training data property, we analyze the parameter differences induced by such properties on an abstract level.

t-SNE (*t-Distributed Stochastic Neighbor Embedding* (Van der Maaten and Hinton, 2008)) is a form of dimensionality reduction which is useful for clustering and visualizing high-dimensional data sets. In particular, the algorithm needs no other input than the data set itself and some randomness.

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

t-SNE Results. As depicted in Fig. 10, t-SNE has produced a well defined clustering for the two image

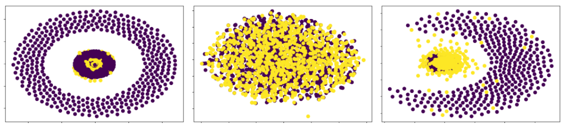


Figure 10: t-SNE visualization of MNIST (left), Census (center) and UTKFace (right) models. Each yellow dot represents a model with property $\mathbb{A}$, each purple dot a $\mathbb{B}$ model.

data sets MNIST and UTKFace: models trained with property $\mathbb{A}$ training data sets (yellow dots) are placed close to the center of the visualization, while property $\mathbb{B}$ models (purple dots) are mostly further from the center. This indicates that the properties, defined in Sect. 5 for MNIST and UTKFace, heavily influence the weights and biases of the trained models. In fact, without any additional information about the parameters or the properties of the underlying training data sets, t-SNE is able to distinguish the models by property with surprisingly high accuracy. Based on these results, one could construct a simple PI adversary $\mathcal{A}^{t\text{-}SNE}$ by measuring the euclidean distance ℓ of a target model from the center of the t-SNE clustering. If ℓ is below a certain threshold for a target model $\mathcal{M}$, $\mathcal{A}^{t\text{-}SNE}$ infers property $\mathbb{A}$, otherwise it infers property $\mathbb{B}$. For MNIST, $\mathcal{A}^{t\text{-}SNE}$ has 86.7% accuracy based on our experiment, while the UTKFace $\mathcal{A}^{t\text{-}SNE}$ has 72.0% accuracy. We stress that these two $\mathcal{A}^{t\text{-}SNE}$ are solely based on the t-SNE visualization of the model parameters, no training on shadow models is needed.

However for Census, t-SNE has not clustered models with different properties of their training data sets apart (see second visualization in Fig. 10). In contrast to the other two data sets MNIST and UTKFace, Census is a tabular data set. It also may be that the properties defined in Sect. 5 have a smaller imminent impact on the weights and biases during training. We leave a more profound analysis of possible reasons for the different behavior of the t-SNE visualization on the three data sets for future work.

7 DISCUSSION

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

Choosing the Right Defense Approach. We have introduced defense mechanisms at different stages of the ML pipeline. Both property unlearning experiments are positioned after the training and before its prediction phase, respectively its publication. In contrast, the preprocessing approach is applied prior to the training. Since most ML algorithms require several preprocessing steps, implementing a defense mechanism based on preprocessing training data could be easily adapted in real-world scenarios. At least for tabular data, our preprocessing experiments (see full paper (Stock et al., 2022)) have shown a good privacy-utility trade-off, especially the artificial data approach. Nevertheless, depending on the organization and application scenario of a ML model, a post-training approach like property unlearning might have its benefits as well. Further experiments could test the combination of both pre- and post-training approaches. Since both of them are not promising to provide the generic PIA defense we aimed for, we assume the combination of both does not significantly improve the defense. Instead, we suggest to focus further analyses on other approaches *during* the training, as laid out in Sect. 8.

Lessons Learned. With our cross-validation experiment in Sect. 5.2, we have shown how PI adversaries react to **property unlearning** in different ways. Some adversaries could still reliably infer training data properties after 15 property unlearning iterations, while other adversaries reliably inferred the *wrong* property after the same process. This shows that it is **hard to utilize a post-training technique** like property unlearning as a generic defense against a whole class of PI adversaries: After all, one needs to defend against the strongest possible adversary while simultaneously being careful not to introduce additional leakage by adapting the target model too much. Depending on the adversary instance, most of our target models clearly show one of these deficiencies after 15 rounds of property unlearning.

Our t-SNE experiment in Sect. 6 shows that at least for image data sets, statistical **properties of training data sets have a severe impact** on the trained parameters of a ML model. This is in line with the LIME experiment, which shows how two PI adversaries with the same objective focus on different parts of target model parameters. If a property is manifested in many areas of a model's parameters, PI adversaries can rely on different regions. This implies that completely pruning such properties from a target model after training is hard to impossible, without severely harming its utility.

8 FUTURE WORK

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

Adapting the Training Process. Another method from a similar area called *fair representation learning* is punishing the model when learning biased information by introducing a regularization term in the loss function during training, e.g., (Creager et al., 2019). As a defense strategy against PIAs, one would need to introduce a loss term which expresses the current property manifestation within the model and causes the model to hide this information as good as possible. In theory, this would be a very efficient way to prevent the property from being embedded in the model parameters. Since it would be incorporated into the training process, the side effects on the utility of the target model should be low.

Post-Training Methods. (Liu et al., 2022) experiment with knowledge distillation (KD) as a defense against privacy attacks like membership inference. The idea is to decrease the number of neurons in an ANN in order to lower its memory capacity. Unfortunately, the authors do not consider PIAs – it would be interesting to see the impact of KD on their success rate.

9 CONCLUSION

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

Apart from the *post-training* defense property unlearning, we have also tested different training data *preprocessing* methods (see full paper version (Stock et al., 2022)). Although most of them were not directly targeted at the sensitive property of the training data, some methods have shown promising results. In particular, we believe that generating a property-free, artificial data set based on the distribution of an original training data set could be a candidate for a PIA defense with very good privacy-utility tradeoff.

ACKNOWLEDGEMENTS

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

REFERENCES

Ateniese, G., Mancini, L. V., Spognardi, A., Villani, A., Vitali, D., and Felici, G. (2015). Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *IJSN*.

Creager, E., Madras, D., Jacobsen, J.-H., Weis, M., Swersky, K., Pitassi, T., and Zemel, R. (2019). Flexibly fair representation learning by disentanglement. In *ICML*.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *TCC*.

Fredrikson, M., Jha, S., and Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*.

Ganju, K., Wang, Q., Yang, W., Gunter, C. A., and Borisov, N. (2018). Property inference attacks on fully connected neural networks using permutation invariant representations. In *CCS*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *IEEE*.

Liu, Y., Wen, R., He, X., Salem, A., Zhang, Z., Backes, M., Cristofaro, E. D., Fritz, M., and Zhang, Y. (2022). ML-Doctor: Holistic risk assessment of inference attacks against machine learning models. In *USENIX Security*.

Mahloujifar, S., Ghosh, E., and Chase, M. (2022). Property inference from poisoning. In *S&P*.

Melis, L., Song, C., De Cristofaro, E., and Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. In *S&P*.

Nasr, M., Shokri, R., and Houmansadr, A. (2018). Machine Learning with Membership Privacy using Adversarial Regularization. In *CCS*.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *ASIACCS*.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Why should i trust you? explaining the predictions of any classifier. In *SIGKDD*.

Rigaki, M. and Garcia, S. (2020). A survey of privacy attacks in machine learning. *Arxiv*.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *S&P*.

Song, C., Ristenpart, T., and Shmatikov, V. (2017). Machine learning models that remember too much. In *CCS*.

Song, C. and Shmatikov, V. (2020). Overlearning Reveals Sensitive Attributes. In *ICLR*.

Song, L. and Mittal, P. (2021). Systematic evaluation of privacy risks of machine learning models. In *USENIX Security*.

Stock, J., Wettlaufer, J., Demmler, D., and Federrath, H. (2022). Lessons learned: Defending against property inference attacks. *arXiv preprint arXiv:2205.08821*.

Suri, A., Kanani, P., Marathe, V. J., and Peterson, D. W. (2022). Subject membership inference attacks in federated learning. *Arxiv*.

Tang, X., Mahloujifar, S., Song, L., Shejwalkar, V., Nasr, M., Houmansadr, A., and Mittal, P. (2021). Mitigating membership inference attacks by self-distillation through a novel ensemble architecture. *USENIX Sec.*

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *JMLR*.

Zhang, W., Tople, S., and Ohrimenko, O. (2021). Leakage of dataset properties in multi-party machine learning. In *USENIX Security*.

Zhang, Z., Song, Y., and Qi, H. (2017). Age progression/regression by conditional adversarial autoencoder. In *CVPR*.