Membership Inference Attacks on Machine Learning Models: Analysis and Mitigation

by

Md Shamimur Rahman Shuvo

Bachelor of Computer Science and Engineering,
Rajshahi University of Engineering and Technology,
Bangladesh, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Faculty of Computer Science

Supervisor(s):	Dima Alhadidi, Ph.D., School of Computer Science, UWINDSOR
	Saqib Hakak, Ph.D., Department of Computer Science
Examining Board:	Kalikinkar Mandal, Ph.D., Faculty of Computer Science, Chair
	Rongxing Lu, Ph.D., Faculty of Computer Science
External Examiner:	Mohsen Mohammadi, Ph.D., Department of Mechanical Engineering, UNB

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

January, 2021

Abstract

Given a machine learning model and a record, membership attacks determine whether this record was used as a part of the model's training dataset. Membership inference attack can present a risk to private datasets if these datasets are used to train machine learning models and access to the resulting models is open to the public. For example, knowing that a certain patient's record was used to train a model associated with a disease can reveal that the patient has this disease. To construct attack models, multiple shadow models are created that imitate the behavior of the target model, but for which we know the training datasets and thus the ground truth about membership in these datasets. Attack models are then trained on the labeled inputs and outputs of the shadow models. There is a desideratum to conduct more analysis about this attack and accordingly to provide robust mitigation techniques that will not affect the target model's utility. In this thesis, we discussed new combinations of parameters and settings, which were not explored in the literature to provide useful insights about the behavior of the membership inference attack. We also proposed and evaluated different mitigation techniques against this type of attack considering different training algorithms of the target model. Our experiments showed that, the defense strategies mitigate the membership inference attack considerably while preserving the utility of the target model. Finally, we summarized and compared the existing mitigation techniques with our results.

Dedication

This thesis is dedicated to my parents, Md Redwanur Rahman and Shahanara Begum who bring all the love to my life.

To my better half Mumu Aktar who sacrificed a lot during my study.

To my lovely sisters, my niece and nephew.

Acknowledgements

All the praise to God, the almighty for granting me the opportunity to write this thesis.

I would like to thank my supervisor, Dr. Dima Alhadidi, to give me the opportunity to work under her supervision. Without her guidance, suggestions, and feedback through out the preparation of the thesis, it would be beyond possible to complete the thesis. I thank my co-supervisor, Dr. Saqib Hakak, for his support and suggestions.

I also thank Dr. Ali Ghorbani for considering me as a member of the Canadian Institute for Cyber Security (CIC). I would also like to thank all the members of CIC.

I would also like to thank my friends Rashedul Alam, Arif Alam, and Mahbubur Rahman for giving me support and suggestions through out the thesis.

I am grateful to my beloved life partner, Mumu Aktar for sacrificing and helping me during my study. Last, but not the least, I thank my parents and other family members for supporting me throughout my life.

Table of Contents

List of Tables

List of Figures

Abbreviations

ANL	Adversarial Noise Layer
API	Application Programming Interface
CNN	Convolutional Neural Network
DP	Differential Privacy
DPSGD	Differentially Private Stochastic Gradient Descent
FFNN	Feed-forward Neural Network
GAN	Generative Adversarial Network
GNL	Gaussian Noise Layer
HE	Homomorphic Encryption
LSTM	Long Short-term Memory
ML	Machine Learning
MLaaS	Machine Learning as a Service
PDF	Probability Density Function
PPML	Privacy Preserving Machine Learning
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SGX	Software Guard Extension
SMPC	Secure Multi-Party Computation
TEE	Trusted Execution Environment
VAE	Variational Auto Encoder

Chapter 1

Introduction

Machine learning (ML) is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions [14]. It has become the base of many internet services ranging from voice recognition, natural language processing, image classification and many more. Many real world companies use machine learning to improve their business through marketing and advertising by analysing user behavior and trends. Most of the internet giant companies like Amazon, Google, and Microsoft provide machine learning as a service (MLaaS) platform to the users lacking machine learning expertise. In MLaaS, a user uploads their own datasets to the server which is processed inside the server and a trained model is returned to the user as a black-box API which is used by the user to help developing their business.

Three different roles are possible in any ML systems. The input party also known as contributors who owns the data, the computation party who performs the required ML tasks, and the prediction party who predicts the new samples as well as help structuring the data based on the model provided by the computation party. In such architecture, the data owner(s) send their collected data to the computation party

to perform the desired ML tasks and formulate a model (also known as a hypothesis function) and delivers the model to the prediction party. The prediction party then predicts the new samples based on the model and output the prediction result to the user. The privacy is naturally preserved if all the three roles are carried out by a single entity, but for different entities, privacy becomes a major concern.

With the increasing popularity of machine learning, ML models are vulnerable to various types of privacy and security attacks such as membership inference attack [74, 78], model inversion attack [21, 22], model extraction/stealing attack [85], property inference attack [2, 23, 52], hyper-parameter stealing attack [64, 91], poisoning attack [84], adversarial attack [8, 27, 43, 66, 67, 89, 96] and many more. Among the attacks on ML models, in this thesis, we focused on analysing membership inference attack and developing mitigation techniques to encourage individual data owners to share their data and help developing a better ML model by preserving the privacy of the data from this attack without compromising the effectiveness of the model.

1.1 Motivation

Membership inference attack determines the presence of a data record in the training data of the model. A successful membership attack can present severe risk to private datasets (e.g. healthcare data and location data) if these are used to train a machine learning model and access to the resulting model is open to the public. For example, if a machine learning model is trained on the data collected from people with a certain disease, by knowing that a patient's data belong to the training data of the model, an attacker can learn the patient's health condition and thus the privacy of the individual can be breached. This is also important for machine learning services such as Google Prediction, as an inability to guarantee the privacy of training data would

preclude a significant number of customers from using their services. Membership inference attack has been successfully conducted on healthcare data [4] and also on location data [69].

Shokri *et al.* [78] was the first to conduct membership inference attack on machine learning models. Shokri *et al.* [78] have investigated the attack in the most difficult setting, where the adversary's access to the model is limited to the black-box queries that return the model's output on a given input. The attack is based on the idea that there are noticeable patterns in outputs which are noticed and evaluated by a machine learning model. Keeping this in mind, Shokri *et al.* [78] used multiple machine learning models (one for each predictive class), known as attack models, to infer the members from the target model. The authors [78] proposed to build multiple shadow models to imitate the behavior of the target model and to generate the data necessary for the attack model training.

Shokri *et al.* [78] made two important assumption to conduct the membership inference attack against machine learning models. First one is to construct multiple shadow models with the same structure as the target model to mimic the target model's behavior. Second assumption is related to the data generation for the shadow model training. Shokri *et al.* [78] used the same distribution as the target model's training data to generate the data to train the shadow models which is eventually relaxed by synthetic data generation.

Salem *et al.* [74] relaxed the assumptions made by Shokri *et al.* [78] to broaden the scope of membership inference attacks against machine learning models. The authors used three different adversarial scenarios based on the design and training data of the shadow models. In the first scenario, Salem *et al.* [74] used a single shadow model along with a single attack model to infer the members but the data generated

for the shadow model has the same distribution as the data used to train the target model which is relaxed in the second scenario. In this scenario, the adversary has no knowledge about the target model's structure or the data distribution. Finally, in the last scenario, Salem *et al.* [74] dropped the idea of using shadow model training for the membership inference attack. The attacker queries the output of the target model with the target data and based on the statistics (such as entropy) differentiates the members from the non-members of the data. This scenario overcomes the training process but relies on threshold choosing method (choose a threshold to take the data after that threshold) for successful and concrete membership inference attack.

Defending against membership inference attacks is an urgent research problem. Several mitigation techniques have been introduced so far against membership inference attacks on machine learning models. Some examples are regularization [78], differential privacy [37, 71], min-max game [61], dropout [74], and model stacking [74]. Some of the present defense strategies rely on a third party to guarantee membership privacy which can easily be breached by the compromised third party. Differential mechanism provides privacy guarantee up to the privacy parameter but imposed a significant amount of classification accuracy loss for protecting large models with high dimensional data. For this, there is a desideratum to conduct more investigations to discover the different reasons behind this attack and accordingly to provide robust mitigation techniques that will not affect the utility of machine learning models. We investigated membership inference attacks under a black-box access scenario in which the adversary can only probe the target model with an input and receive the corresponding predictions.

1.2 Objectives

The main objectives of the thesis can be summarized as follows:

– To provide useful insights about the membership inference attack on machine learning models.

– To characterize the membership inference attack on machine learning models from different perspectives and combination of training algorithms.

– To develop the defense techniques against this attack which will maintain both the utility of the model as well as the privacy of the members taking part in training the model.

1.3 Our Contributions

My contributions [79] in this thesis can be summarized as follows:

– We discussed new combinations of parameters and settings, which were not explored in the literature to provide useful insights about the behavior of the membership inference attack. We conducted an empirical study to characterize the attack from different perspectives: number of shadow models, number of attack models, confidence thresholds, and different combinations of training algorithms for the different models (the target model, the shadow models, and the attack model) included in the attack.

– We studied the effect of Gaussian noise layers of Lasagne [12], which is a light-weight Python library, to mitigate membership inference attacks for target models trained with feed-forward neural networks.

– We studied the effect of regularization to mitigate membership inference attacks for target models trained with logistic regression, To the best of our knowledge, this thesis is the first to conduct such an experiment.

– We studied the effect of Gaussian noise layers of Lasagne [12] and Adversarial Noise layer (ANL) (regularization) [99] to mitigate membership inference attacks for target models trained with Convolutional Neural Network (CNN).

– We proposed and evaluated the use of the exponential mechanism [49] for all three types of training algorithms (feed-forward neural networks, convolutional neural networks, and logistic regression) of the target model to mitigate the membership inference attack.

– We conducted a comparative study between some existing mitigation techniques and our results in terms of the target model training accuracy, target model testing accuracy, and attack accuracy. The comparison of different existing countermeasures against membership inference attacks has a great relevance for the design of future models.

1.4 Thesis Organization

Rest of the thesis is organized as follows. In Chapter 2, we discussed the related work. We discussed the different attacks on machine learning models, different mitigation techniques against the membership inference attack and some privacy-preserving techniques to preserve the privacy while model generation and prediction. We detailed the membership inference attack as described by Shokri *et al.* [78] in Chapter 3. The analysis of the membership inference attack is presented in Chapter 4 which is followed by the results of our proposed mitigation techniques together with the comparison study in Chapter 5. In fine, we conclude our thesis mentioning about the future scope of the research in Chapter 6.

Chapter 2

Related Work

In this chapter, we first detailed different types of attacks against machine learning. Then we summarized the related work to the membership inference attack together with the existing mitigation techniques. At the end, we summarized the privacy-preserving mechanisms for the machine learning models during computation and prediction.

2.1 Attack against Machine Learning Models

Machine learning models are vulnerable to various types of attack. Different types of privacy or confidentiality attacks has been successfully conducted on ML models. Among them, model inversion attack [21, 22], model extraction attack [85], property inference attack [2, 23, 52], hyper-parameter stealing attack [64, 91], poisioning attack [84], and adversarial attack [8, 27, 43, 66, 67, 89, 96] are noteworthy.

Model Inversion Attack. In model inversion attack, an attacker tries to infer sensitive attributes of given data instances from a released model and also the instance's non-sensitive attributes. Fredrikson *et al.* [22] successfully conducted a model inversion attack in biomedical settings. This attack is able to use black-box access to

prediction models in order to estimate aspects of someone's genotype. Model inversion attack has also been investigated on commercial machine learning as a service (MLaaS) APIs [21] to recover images from a face recognition model.

Model Extraction Attack. Model extraction attack also known as model stealing attack recovers the model parameters via black-box access to the target model. Tramèr *et al.* [85] explored a model extraction attack that finds the parameter of a model given its predictions (confidence values).

Hyper-parameter Stealing Attack. Another form of model extraction or stealing attack is the hyper-parameter stealing attack [64, 91] where an attacker's goal is to know the different hyper-parameters of neural networks used during training the model. Effective adversarial examples can be generated against the black-box model by revealing the internal information of the target model [64]. Hyper-parameter stealing attack does not only target neural networks but also other types of ML algorithms like logistic regression and support vector machine [91].

Property Inference Attack. The aim of this attack is to infer certain property of information from the target model [2, 23, 52]. A meta-classifier has been built to infer useful information from other classifiers to obtain the information about the training sets for the target classifier [2]. Melis *et al.* [52] has exploited property inference attack on collaborative learning while Ganju *et al.* [23] inferred the global properties of the training data from a white-box fully connected neural network.

Poisoning Attack. Noise is added to the data points in a controlled way to mislead the training process during poisoning attack [84]. Perturbed inputs are used in adversarial training to increase the robustness of the model built with small datasets. But for models with large datasets, adversarial training converges to a degenerated global minimum. Adversarial training generally defends the model from the attack.

But due to poisoning attack, the adversarial training generates weak perturbations for models built with large datasets which failed to defend the model [84].

Adversarial Attack. Perturbing an input in a specific way that causes a machine learning model to make a false prediction is known as adversarial attack [8, 27, 43, 66, 67, 89, 96]. Autonomous driving and voice recognition system face severe risks due to this type of attack.

Membership Inference Attack. Membership inference attack determines the presence of a data sample inside a certain dataset. Homer *et al.* [33] developed a method to infer the members in biomedical settings specially on genomic data which is further studied in [4, 19]. Comparing the summary statistics like mean and standard deviation with the target database, the presence of the data in the database can be determined [33]. Log-likelihood ratio test has been used as the statistical testing method. Backes *et al.* [4] and Hagestedt *et al.* [28] used the similar membership inference attacks to detect membership from MicroRNA and DNA methylation data, respectively. Mobility data [69, 70] can also be subject to membership inference where the attacker infers the presence of a user's location used for computing a model from a location dataset.

2.2 Membership Inference Attack against Machine Learning Models

Shokri *et al.* [78] conducted the first membership inference attack against machine learning models to arbitrate the presence of a data sample during training a model from a black-box target model. To construct the attack model, the attacker creates multiple shadow models similar to that of the target model to replicate the behavior of the target model and also to generate the data for training the attack

model. The data to train the shadow model are assumed to be of the same distribution as one of the target model. The authors [78] built multiple attack models each representing a predictive class of the target model. The shadow models give probability score vectors which are queried with the dataset used for training the particular shadow model. The data are labelled "in" if used for training and labelled "out" if used for testing. The probability score vector along with the label "in/out" are used to train the attack model. As the final class label of the attack model are either "in" or "out", the attack model is a binary classifier which infers the members of the target model based on the probability score vector of the target model output. The authors [78] used multiple shadow models and class-specific attack models for the successful membership inference attack. We experimentally analysed the usefulness of having multiple shadow models for the successful attack. We also experimentally demonstrated the difference of having class-specific attack model and class-independent attack model for the success of this type of attack. In a class-specific attack model, the data points are separated for a particular class. For example, if we have a dataset consisting of "n" classes, then a class-specific attack model will built "n" models for each individual class. But for a class-independent attack model a single model will be built with the "n" classes. We demonstrated how these two type of models affect the membership inference attack.

Salem *et al.* [74] broaden the scope of the membership inference attack by relaxing the assumption of Shokri *et al.* [78] in terms of both model and data independent adversary in a black-box target model. First of all the authors [74] relaxed the assumption on the number of shadow models by using a single shadow model to generate the data for the attack model training. But using a single shadow model provides less data to the attack model to be trained on. As a result, the attack accuracy becomes less than when using multiple shadow models. Later, Salem *et al.* [74] showed that the data need not to be of the same distribution as the target

data and the structure can be different for training the shadow model. Finally, they proposed the attack without any shadow training scheme by relying only on the output of the target model queried with target data points.

Membership inference attack has also been conducted on federated learning [62, 51, 87]. Nasr *et al.* [62] conducted the attack on white-box deep learning models to infer the data records. The attacker utilized the privacy vulnerability of the stochastic gradient descent of the deep learning model to collect the data sample's feature and conduct the membership inference attack. In collaborative machine learning, participants share their model parameter with each other to built a joint model. Attack can be successfully conducted by any adversarial participants to infer the data points in other's training set and also the property of any subset of data can be inferred [51].

Truex *et al.* [87] showed the vulnerability of membership inference attack in collaborative learning if the adversary is one of the participants in the learning process. Truex *et al.* [87] comprehensively studied the membership inference attack to demystify two different perspectives: developing a generalized black-box membership inference attack model and showing the importance of the model choice. The showed the transferability of the attack model of the membership inference attack. Irolla and Châtel [35] showed that the confidence level interval of the "in sample" and the "out sample" has no or a little impact on the success of the membership inference attack. Membership inference attack has also been conducted using an imbalanced dataset and on an adversarial machine learning model [86]. The minority class of the imbalanced dataset is more vulnerable and increases the risk of the membership inference attack than the majority class. It is also shown [86] that differential privacy presents many trade-offs as a mitigation technique to membership inference risk.

Given a distribution of any original dataset, realistic sample can be generated using a generative model. Hayes *et al.* [30] showed that the state-of-the-art Generative Adversarial Networks (GANs) are vulnerable to the membership inference attack in both white-box and black-box settings. The authors [30] successfully attacked a generative adversarial network (GAN- combination of discriminative and generative model) and detect the overfitting by learning the statistical differences in the data distribution. Hilprecht *et al.* [32] studied the attack on two different architectures namely Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) on standard image data which outperformed the previous attack presented by Hayes *et al.* [30]. Effectiveness of the membership inference attack against GANs largely depends on the model type and training configuration of the model [10]. Chen *et al.* [10] experimentally analysed the effectiveness of the attack on deep GANs using image, location and medical data.

Large data in social media helps in developing social media analytic model which can also be vulnerable to the membership inference attack. Liu *et al.* [44] inferred the members of a training dataset by detecting the prediction behavior of the publicly available social analytic model with a mimic model that shows similar predictive behavior as the public model. Zhang *et al.* [102] investigated the algorithmic fairness issue in membership inference attack by formalizing the vulnerability disparity (VD) notation that quantifies the attack difference in different demographic groups. Long *et al.* [47] studied the effect of vulnerable instances on the membership inference attack and developed a new generalized membership inference attack. Overfitting and compromised adversary in collaborative learning are not the main factor for successful membership inference attack, vulnerable data in the training dataset are also responsible for the attack. It should be mentioned that, most of the membership inference attacks were conducted on classification models [46, 47, 61, 62, 74, 78, 98].

2.3 Mitigation Techniques of Membership Inference Attack

Membership inference attack breaches the privacy of the data taking part in training a machine learning model. Protecting the machine learning model from this attack is a major research problem. In the following, we summarize the defence techniques introduced so far against this attack.

2.3.1 L2-Regularizer

Overfitting in machine learning model is a common reason but not the only reason for a successful membership inference attack. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. There are a number of techniques such as $L1-$regularization, $L2-$regularization, and cross-validation to overcome the overfitting problem of ML models.

Shokri *et al.* [78] used the conventional $L2-$regularizer as a defence technique to overcome the membership inference attack on machine learning models trained using a neural network. $L2-$regularizer adds *"squared magnitude"* of coefficient $\lambda \sum_i \beta_j^2$ as the penalty term to the model's loss function, where β_j are the model's parameters and λ is the regularization parameter. Larger regularization parameter gives larger regularization effect during training. Shokri *et al.* [78] evaluated the effect of $L2-$regularization on two different target models of which one is constructed using cloud-based "Machine Learning as a Service" platform and the other one is with a target model trained with a neural network. The evaluation result by using $L2-$regularizer to mitigate the membership inference attack was not effective enough. It reduced the model's performance on the test data. To avoid the per-

formance degradation of the model and to increase the privacy of the data, other regularization methods are needed. Our proposed method of Gaussian noise layer in neural networks overcomes this limitation of performance degradation of the model as well as prevents the data leakage by the attack model.

2.3.2 Min-Max Game

Min-max is a kind of backtracking algorithm that is used in making a decision. The benefit of min-max game theory is minimizing the possible loss of a scenario while maximizing the gain of the same.

Nasr *et al.* [61] studied the min-max privacy game between the membership inference attack and the defence technique of the target model. To achieve the goal, Nasr *et al.* [61] designed an optimization problem which minimizes the classification error of the target model (high utility) as well as the maximum gain of the membership inference attack against it (high privacy). An adversarial training process has been introduced to train the model in a similar way as that of generative adversarial networks [26] and other adversarial training processes for machine learning models [13, 15, 41, 54, 55, 63]. The gain of the attack has been added as a regularizer for the target model to protect the data privacy of the model. The trade-off between the classification error and the membership privacy can be controlled using a regularization parameter in this method.

Min-max optimization technique against membership attack formalized a joint privacy and classification objective where the inner maximization finds the strongest inference attack model against a given classification model and the outer minimization finds the strongest defence for that classification model. An external parameter λ controls the trade-off between privacy and accuracy of the model and also acts as a regularizer for the model. Two different models are trained alternatively in each

epoch to find the best response for the nested optimization problem which consumes time during training.

2.3.3 Dropout

Dropout is a way of regularization that drops a neuron with a certain probability during the training of a neural network in each iteration [81]. By approximating a large number of neural networks, training with different architectures in parallel, this method regularizes a neural network. Dropout is very much effective in reducing the overfitting problem of a neural network built model.

Salem *et al.* [74] explored the use of dropout as a mitigation technique against membership inference attack for a model trained with a neural network. The dropout has been added for both the input and hidden layer of the neural network to delete a specified portion (dropout ratio) of the edges in each training iteration randomly. Their empirical results showed the effectiveness of the target model in reducing the overfitting level which has been calculated by subtracting the original target model's overfitting level from the dropout-defended target model's overfitting level. Effective dropout results in larger reduction of overfitting level and thus protecting the target model from the membership inference attack.

Reducing overfitting level largely depends on the dropout ratio. Larger dropout ratio reduces the attack performance more than the smaller dropout ratio which also affects the performance of the model's utility. With large dropout ratio, lower is the attack performance as well as the utility of the target model. It is required to mediate the dropout ratio to maintain a trade-off between the utility of the target model and privacy of the training data. It is to be mentioned that, dropout does not work well in preserving the privacy of the target model from the membership inference attack for a small featured dataset.

2.3.4 Model Stacking

Salem *et al.* [74] also used another defence mechanism for preserving the privacy of the target model. Dropout is only limited to the target model trained with a neural network. For target models trained with other algorithms, model stacking can be used. Model stacking is a type of ensemble learning algorithm where multiple weak classifiers are organized in a hierarchical way. The output of one classifier is used as the input of the classifier in the next level and finally the output of the classifier in the last level is used as a final model for the classifier. Figure 2.1 shows a simple architecture of model stacking.

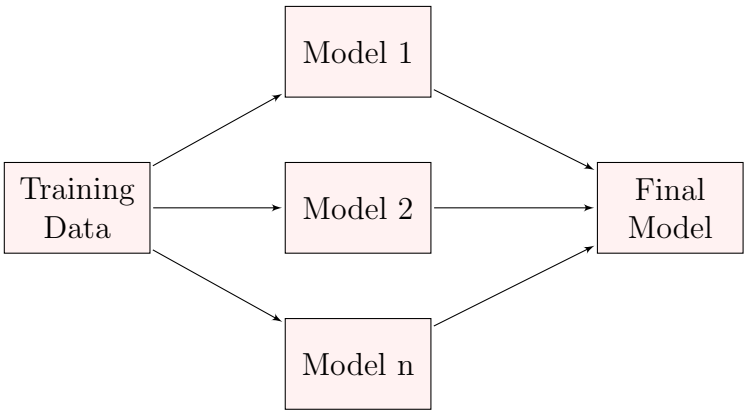

Figure 2.1: Model Stacking

Salem *et al.* [74] used a two-level model stacking architecture to preserve the privacy of the target model from the membership inference attack where the first level takes the original data sample as the input and the second level takes the posteriors of the first level as the input. Finally, the output of the second level is the resultant model. The idea behind this defence strategy is that, different subsets of data are used to train different parts of the target model which helps reducing the overfitting possibility of the target model. Salem *et al.* [74] first applied the input on each of the two models to get the posteriors which are concatenated and applied to the third model to get the final output. Salem *et al.* [74] trained the

models of the first layer with disjoint set of data (i.e. no model were trained with same data points to maximize the prevention of overfitting). Their [74] experimental results proved the effectiveness of model stacking against the membership inference attack. Though model stacking is effective against the membership inference attack, the time complexity is high for this type of training as multiple models are trained to get a final model. We proposed and evaluated the use of noise layers at the time of training the target model such that not to increase the training time of the target model as well as to protect the target model from the membership inference attack.

2.3.5 Differential Privacy

The classical method for privacy-preserving machine learning is the differential privacy [17]. An algorithm is differentially private if the presence of any individual's data can not be found in the original dataset by looking at the output of the algorithm [95]. Many privacy attacks including linkage attack [60, 83], reconstruction attack [13] and differencing attack [33] can achieve mathematical provable guarantee of privacy using differential privacy [18]. It guarantees to protect only the private information, but not the general information. Differentially private algorithms rely on random noise to make the data noisy and imprecise to help preserving the privacy of the data [16]. There are two different methods, noise and exponential mechanism [49] to achieve differential privacy.

Rahman *et al.* [71] studied the impact of the membership inference attack against a differentially-private deep learning model. A composite function has been learnt by a deep neural network from the inputs after passing through different nonlinear transformations on the inputs in multiple layers. Rahman *et al.* [71] bounded the sensitivity of the component functions in each layer and noise has been added to the bounded-sensitivity functions to achieve a differentially private deep model. The

diffierentially private deep model has been composed of two components. One is the Differentially Private Stochastic Gradient Descent (DPSGD) to achieve a differentially private version of the stochastic gradient descent algorithm and another is the moments accountant [1] where privacy has been bounded by a budget.

In DPSGD, Rahman *et al.* [71], clipped the gradients to achieve maximum $L2-$norm bound before updating the parameter which prevents any data to dominate the training and to overcome the overfitting of the model parameters. After that, Gaussian noise has been added to the clipped gradients before updating the parameter to achieve the differential privacy. Privacy of the model has been bounded by the noise scale and the clipped threshold for the gradients in DPSGD by the moments accountant.

Differentially-private deep models provide privacy protection by considerably sacrificing the model utility [71, 86]. Noise has been added to the objective function [9, 36, 40] to learn a model in a differentially private system. Noise can also be added to the gradient descent or stochastic gradient descent in each iteration to minimize the objective function [1, 5, 80, 92, 100] of the model to increase the privacy. Shokri and Shamitkov [77] designed a differentially-private approach for deep neural network in the collaborative learning.

Differential privacy guarantees a privacy up to the privacy parameter ϵ. The lower the privacy budget ϵ, the lower the information leakage and the higher the privacy guarantee. Most of the differential privacy mechanisms do not include utility in the design objective of the privacy mechanism. As a result, this might impose a significant loss of utility for protecting a large model with high dimensional data for small privacy budget ϵ.

2.3.6 MemGuard

Jia *et al.* [37] evaluated a defence method named "MemGuard" to guarantee the formal utility-loss against the membership inference attack in black-box settings. Jia *et al.* [37] explored the use of a noise vector to turn the confidence score vector of the model into an adversarial example that misleads the attacker's classifier.

MemGuard is a two-phase process. In the first phase, "MemGuard" finds a noise vector to turn the confidence score vectors into adversarial examples. This phase relies on a third party to calculate the noise vector. This third party trains a classifier for membership inference to craft the noise vector based on it's own classifier which aims to mislead the attacker's classifier as the noise vector misleads it's own classifier in inferring the members. In the second phase, "MemGuard" adds the noise vector to the true confidence score vector with certain probability such that the expected confidence distortion is bounded by a budget. Jia *et al.* [37] achieved two goals through "MemGuard". Firstly, they mislead the attacker's classifier to ensure privacy of the data and secondly, they ensure the utility of the model.

"MemGuard" relies on a third party to calculate the noise vector to protect the target model from the membership inference attack. But if the third party is compromised, it would not be possible to ensure the privacy of the data taking part in training a model. It also relies on the parameter **"ϵ"** to control the trade-off between membership privacy and confidence score vector distortion which is dataset dependent. Jia *et al.* [37] defined a threshold for the inference accuracy for various classifiers and modified ϵ based on the threshold. Our proposed mitigation strategy has no such assumption about the threshold and it does not rely on any third party to calculate the noise for the target model.

2.3.7 Weight Normalization

Weight normalization is a method that normalizes the weights of the layers of a neural network to decouple the norm of the weight vector from its direction [75]. Hayes *et al.* [30] explored the possibility of weight normalization as a defence mechanism against the membership inference attack on generative models. The underlying distribution of a dataset are estimated to generate realistic samples according to the distribution in a generative model. This model includes the distribution of data itself to tell how likely a given sample would be. Generative models are much simpler than generative adversarial networks (GANs) as this model can assign a probability to a sequence and also can generate new data instances.

Hayes *et al.* [30] first explored the membership inference attack against generative models and also provided a mitigation technique against the attack through weight normalization. Overfitting has been detected by a combination of discriminative and generative model and the members of the training dataset has been recognized using the discriminator's capacity to learn statistical differences in the distributions which infers the members of the training dataset.

Weight normalization has been used as a defense against the membership inference attack on generative models [30]. Weights has been decoupled the length of the weights from their direction which has been applied to all layers of the generator and discriminator of the target model to minimize the inference attack. Their empirical results showed the improvement of weight normalization over random guessing but it often showed instability in training. Sometimes the discriminator has been outperformed by the generator and vice-versa. Hayes *et al.* [30] also explored the use of dropout as a mitigation technique against the attack on generative models which has been proved effective than weight normalization but consumed more time during training the model.

The mitigation techniques adopted against membership inference attack are summarized in Table 2.1

Authors	**Target Model Structure**	**Target Data Distribution**	**Number of Shadow Models**	**Defence Technique**
Shokri *et al.* [78]	Unknown	Known	Multiple	L2-regularization
Salem *et al.* [74]	Unknown	Unknown	Single	Dropout, Model stacking
Rahman *et al.* [71]	Known	Known	Multiple	Differential Privacy
Nasr *et al.* [61]	Known	Known	-	Min-Max Game
Jia *et al.* [37]	Known	Known	-	MemGuard
Truex *et al.* [86]	Known	Known	Multiple	Differential Privacy
Hayes *et al.* [30]	Known	Known	-	Weight Normalization, Dropout

Table 2.1: Membership Inference Attack Mitigation Techniques

The existing defence mechanisms suffer from either privacy-utility trade off or rely on a third party. Any compromised third party can easily infer the members of the training data of the target model. For this, we proposed and evaluated three different methods for three types of models which will not depend on any third party rather will give the trade off between utility and privacy.

2.4 Privacy-preserving Machine Learning

Privacy-preserving machine learning is another relevant work domain. Several techniques has been introduced so far to protect the data at the computation and the prediction stages of ML systems. Among them Homomorphic Encryption (HE) [31], Secure Multi-Party Computation (SMPC) [72], Differential Privacy (DP) [77], and

Trusted Execution Environment (TEE) [34] using Intel Software Guard Extension (SGX) [48, 76] are noteworthy.

The encryption method that allows encrypted data to be processed and manipulated is the Homomorphic Encryption (HE). Any third party can be able to apply functions on encrypted data without revealing the original data and when decrypted it behaves as if the function is applied on the original data. Gilad-Bachrach *et al.* [24] developed a method named "CryptoNets" to securely predict from a neural network on a cloud-base service. In CryptoNets, data owner sends their data to the service provider in an encrypted form to ensure confidentiality of the data. The service provider applies the neural network to the encrypted data and sends the prediction to the data owner in the same format (i.e. encrypted prediction is sent to the data owner which can be decrypted by the data owner to get the desired prediction).

Wang *et al.* [93] experimentally proved the practicality and effectiveness of homomorphic encryption in training a neural network in collaborative learning. Wang *et al.* [93] trained distributed word vectors on encrypted data from multiple participants to ensure privacy-preserved training of a neural network using HE.

Hesamifard *et al.* [31] evaluated a new method, "CryptoDL" for both training and prediction of deep neural network using HE. They experimentally proved the feasibility of training and predicting the data in an encrypted form and also sharing the predicted result in an encrypted form through "CryptoDL" in a machine learning as a service platform.

Using Secure Multi-Party Computation (SMPC), multiple parties jointly computes an arbitrary function over their inputs, keeping their individual data private [104]. Bost *et al.* [6] studied SMPC to securely perform training as well as classification

on biomedical data. The authors identified the efficient set of operations for the encrypted data to ensure the funtionality and security of the private data.

Mohassel and Zhang [57] developed "SecureML" to ensure privacy preserved machine learning training and prediction using stochastic gradient descent. They developed two party computation (2PC) model to securely train various models on the joint data shared by different data owners. "SecureML" proposes MPC-friendly alternatives to non-linear functions.

"DeepSecure" is another framework proposed by Rouhani *et al.* [72] for privacy-preserved deep learning. "DeepSecure" used Yao's Garbled Circuit (GC) protocol [97] to compute secure deep learning. In this method, neither the data owners nor the cloud server are willing to reveal any of their information. SMPC has been implemented on different algorithms namely linear regression, logistic regression and neural networks.

A secure area of the main processor to execute code with high level of trust from the surrounding environment and provide confidentiality and integrity is the Trusted Execution Environment (TEE). Intel provides Software Guard Extension (SGX) [48, 76] enclaves to securely execute the code. Hunt *et al.* [34] explored "Chiron" to privately train a neural network using TEE. "Chiron" conceals the training data from the service provider into SGX enclaves to perform operations and give black-box access of the trained model to the user without revealing the training algorithm or model structure.

Hybrid cryptographic framework has been used [45, 73] to preserve the privacy of machine learning. In the hybrid method, different individual methods are combined to form a new protocol. Liu *et al.* [45] proposed "MiniONN" with SMPC and HE to classify a neural network. Liu *et al.* [45] transformed the neural network to an

oblivious neural network to preserve the privacy of the predictions with considerable efficiency. Sadat *et al.* [73] trained linear regression as well as logistic regression models using TEE (Intel SGX) and somewhat HE in a collaborative learning environment to preserve the privacy of medical data. The privacy-preserving machine learning techniques are summarized in Table 2.2

Author	**ML Algorithm**	**ML Model**	**Adopted Technique**
Hesamifard *et al.* [31]	Neural Network	Training and prediction (CryptoDL) in MLaaS	HE
Bost *et al.* [6]	Classification	Prediction in MLaaS	SMPC
Liu *et al.* [45]	Neural Network	Classification	Hybrid (SMPC and HE)
Mohassel and Zhang [57]	Linear Regression, Logistic Regression, Neural Network	Training and prediction (SecureML) in MLaaS	SMPC
Gilad-Bachrach *et al.* [24]	Neural Network	Prediction (CryptoNets) in MLaaS	HE
Rouhani *et al.* [72]	Deep Neural Network	Prediction (DeepSecure) in MLaaS	SMPC
Hunt *et al.* [34]	Neural Network	Training (Chiron) in MLaaS	TEE
Sadat *et al.* [73]	Linear Regression, Logistic Regression	Training in the collaborative setting	Hybrid (TEE and HE)
Wang *et al.* [93]	Neural Network	Training in the collaborative setting	HE

Table 2.2: Privacy Preserving Machine Learning Techniques

2.5 Conclusion

We detailed the existing literature about the membership inference attack on machine learning models in this chapter. We also discussed the mitigation techniques applied against this attack and their limitations in mitigating the attack. We presented some

other privacy and security attacks on machine learning models. In the last part of this chapter, we discussed the privacy preserving machine learning techniques.

Chapter 3

Membership Inference Attack

In this chapter, we detailed the membership inference attack model as described by Shokri *et al.* [78]. We summarized the different possible ways to generate the data for the shadow model training and also the way to generate the data for the attack model training.

Membership Inference Attack. Machine learning models show different behavior with the data they see for the first time than the data they are trained on. An adversary exploits this behavior of ML models to construct an attack model which infers the members of a training dataset based on the output of the target model. Shokri *et al.* [78] constructed multiple shadow models to mimic the target model and used the output of the shadow model to train the attack model.

The target model has its private dataset that contains the labeled data records, (x_i, y_i) where x_i are the features of the data point and y_i are the true outputs of the data point. The input of the target model is x_i whereas the output from the target model is the probability vector. The class with the highest probability is chosen to be the predicted label for the data record. As we mentioned earlier, shadow models are created to mimic the target model. Shadow models also give similar output like

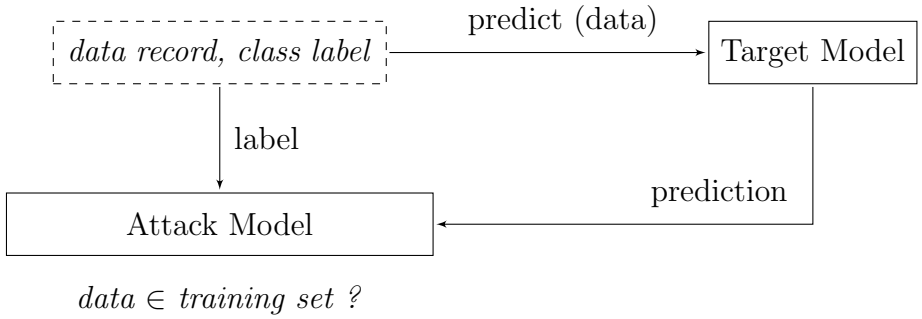

Figure 3.1: Overview of Membership Inference Attack [78]

the target model. However, the shadow models and the target model are trained using datasets that are disjoint from each other (i.e., $D_{target} \cap D_{shadow_i} = \emptyset$).

The attack models are trained using the inputs and outputs of the shadow models. To construct the attack models training data, the training set of the shadow models are queried and labeled "in" and the test dataset of the shadow models are also queried and labeled "out". As a result the training dataset of the attack model contains the output probabilities of the trained dataset with a label "in" or "out". We used the output of the target model to infer the members using the attack model. The overview of the membership inference attack is shown in Figure 3.1.

To understand the membership inference attack, we need to discuss the following three models:

1. Target model
2. Shadow model
3. Attack model

3.1 Target Model

The target model captures the relationship between the content of the data records and their true labels. The target model takes the data records as input and after training outputs the prediction vector of probabilities. These probabilities are termed as confidence values in our work. The class with the highest confidence is chosen to be the predicted label for the data record.

Let us suppose, D_{Target}^{Train} be the private training dataset of the target model, f_{Target} where (x_i, y_i) are the labeled data records. In this labeled data records, (x_i) represents the input to the target model whereas (y_i) represents the true label of the data record which takes the values from a set of classes of size C_{Target}. The output of the target model, f_{Target} is a vector of probabilities of size C_{Target} where the elements are in the range from 0 to 1 and the sum of the vector is equal to 1.

The accuracy of the target model is measured by how the model predicts the labels of other data records from the same population. The attacker has black-box query access to the target model to obtain the prediction vector for any record. The attacker also knows both the input and output formats. The attacker may have some background knowledge about the population from which the target model's training dataset was drawn. Alternatively, the attacker may know some general statistics about the population. Membership attack exploits the idea that machine learning models behave differently on the data they were trained on versus the data they see for the first time. The goal of the attacker is to construct an attack model that can recognise the behaviour differences of the target model to distinguish members from non-members of the target model's training dataset based on the target model's output.

3.2 Shadow Model

The shadow models are created to overcome the challenge of training the attack model to distinguish the members from non-members of the target model's training dataset. These shadow models are created to mimic the target model's behavior and to create the data required for the attack model's training. To train the attack model, multiple shadow models are built to behave similarly as the target model.

The adversary creates n number of shadow models $f^i_{Shadow}()$ where each shadow model i is trained on dataset D^i_{Shadow}. The adversary first splits its dataset D^i_{Shadow} into two sets D^{iTrain}_{Shadow} and D^{iTest}_{Shadow} such that,

$$D^{iTrain}_{Shadow} \cap D^{iTest}_{Shadow} = \emptyset \tag{3.1}$$

Adversary then trains each shadow model f^i_{Shadow} using the training set D^{iTrain}_{Shadow} and test the same using D^{iTest}_{Shadow} set of the data. The dataset, D^i_{Shadow} for each shadow model i follows the same format and distribution of the target model's training dataset D_{Target} which is generated using one of the methods described in the later part of this section. Shokri *et al.* [78] considered the worst case for the adversary such that the dataset used for training the shadow models are disjoint from the private dataset used to train the target model such that,

$$\forall i, D^{Train}_{Shadow^i} \cap D^{Train}_{Target} = \emptyset \tag{3.2}$$

The attack model is trained to recognize differences in shadow models' behaviour when these models operate on inputs from their own training datasets versus inputs they saw for the first time. The more shadow models, the more accurate the attack model will be. As a result, multiple shadow models will provide more data to the attack model for training.

Data Generation for Shadow Models. The adversary needs training data distributed from the same distribution as the target model's training data to train the shadow models. Shadow data can be generated using different methods mentioned below:

1. Model-based data generation
2. Statistics-based data generation
3. Noisy real data

Details of these methods are described below:

Model-based Data Generation. In this method, the adversary has no knowledge about the real training data and also does not have any statistics about the distribution of the target model's training data. The concept is to make a dataset with the data records that are classified with high confidence by the target model will have similar statistics as that of the target model's training dataset and will be a good set for the shadow models. There are two phases in model-based data generation.

1. **Search:** This phase is to find the data records classified with high confidence by the target model. This is an iterative phase where the "Hill-climbing" algorithm [88] can be used for data generation.
2. **Sample:** The phase collects the data after the search phase. This is a repeated phase which runs until the desired number of samples are collected for the shadow model's data.

Algorithm 1 shows the pseudocode for the model based data generation proposed by Shokri *et al.* [78]. The adversary first fixes the class c for the class of the synthetic data. The first phase of the data generation is searching the data records classified

with high confidence which is an iterative process. For this we first initialize a random record X, sampling the value for each feature uniformly at random around the possible values of that particular feature. The initialized record will be accepted if and only if it increases the probability of being classified by the target model as class c. This is the objective of the hill-climbing algorithm.

Algorithm 1 Model-based Data Generation [78]

```
1: procedure GENERATE(class : c)
2:     X ← Rand()                                   ▷ Initialize a record
3:     y_c^* ← 0
4:     j ← 0
5:     k ← k_max
6:     for i = 1 to i_max do
7:         y ← f_target(X)                          ▷ Query the target model
8:         if y_c ≥ y_c^* then                      ▷ Accept the record
9:             if y_c > conf_min and c = argmax(y) then
10:                if rand() < y_c then
11:                    return X
12:                end if
13:            end if
14:            X^* ← X
15:            y_c^* ← y_c
16:            j ← 0
17:        else
18:            j ← j + 1
19:            if j > rej_max then                  ▷ Consecutive rejects
20:                k ← max(k_min, ⌈k/2⌉)
21:                j ← 0
22:            end if
23:        end if
24:        X ← Rand(X^*, k)                         ▷ Randomize k features
25:    end for
26:    return ⊥                                     ▷ Failed to generate data
27: end procedure
```

A new record is proposed by changing k randomly selected features of the latest accepted record X^* in each iteration. Proposing new records are done by flipping the binary features or features of other types are resampled for new features. When

subsequent rejection of the proposed data points, feature k is initialized to k_{max} which is then divided by 2 to control the search for the new record around an accepted record. The minimum value of k is set to k_{min} to control the speed of the search for new records with potentially higher classification probability y_c.

When the target model is confident enough in predicting the class of a particular data record belonging to class c, then the second phase of the data generation (i.e. the sampling phase) starts. This is ensured by the probability (y_c) predicted by the target model of a proposed data record that belongs to class c becomes larger than the probabilities of all other classes and a given threshold $conf_{min}$. The record is selected for the new dataset with probability y_c^*. In case of failure in selecting data, the process is repeated until a record is selected for the new dataset.

The main drawback of this process is that, the adversary needs to explore the space of possible inputs efficiently and have to discover the inputs classified with high confidence by the target model. For high resolution images this process might not work. It will not also work on models with complex classification tasks.

Statistics-based Data Generation. Another method of data generation is the statistics-based data generation where data is generated based on a sample data that reflects the same statistical properties as the original data sample [20]. The adversary knows some statistical information about the population from where the training dataset for the target model was drawn. The adversary may also know the marginal distribution of different features of the target models' training dataset.

Noisy Real Data. In this method of data generation, the adversary has access to some data of the target model's training dataset. Based on those data, the adversary creates a new dataset by flipping the randomly selected features of the accessible data which is the noisy real data.

We explored the statistics-based data generation to generate the data for the shadow model training in most of the cases by sampling the value of each feature from its own marginal distribution. The attack models are very much effective in this case. We also explored the model-based data generation by changing the confidence threshold ($conf_{min}$). The resulting attack gives variable attack accuracy depending on the threshold.

3.3 Attack Model

The attack model is a collection of models, one for each output class of the target data. To train the attack model, Shokri *et al.* [78] used multiple shadow models, which behave similarly to the target model. However, for each shadow model, we know, if a given record is in the training set "in" or in the testing set "out". That is why supervised training on the outputs of shadow models is used to teach the attack model how to distinguish the output of shadow models on members of the training datasets from the output of non-members.

Let, D_{Attack}^{Train} be the training dataset of the attack model, which contains labeled data records (x_i, y_i) together with the probability vector generated by the shadow model for each record x_i and "in" if x_i is used for training the shadow model or "out" if x_i is used for testing it. The attack model's input is composed of a correctly labeled record and a prediction vector of probabilities generated by the target model for the corresponding record. Since the goal of the attack is decisional membership inference, the attack model is a binary classifier with two output classes, "in" and "out".

Data Generation for Attack Model. The training data for the attack model comes from the inputs and outputs of the shadow models. Each of the shadow

models are queried using all the records of its own training dataset as well as with the disjoint test dataset of the same size to get the output. First of all, a particular shadow model is queried with its training dataset to get the output vectors which are labelled "in" and added to the attack model's training dataset. Then the same shadow model is queried with the test dataset disjoint from its training dataset to obtain the output vectors which are labelled "out" and also added to the training set of the attack model. The same process is repeated for all the shadow models to construct the training dataset of the attack model. This dataset reflects the black-box behavior of the shadow models on their training and test data.

Let, $(x, y) \in D_{Shadow}^{Train^i}$ be the training dataset of the i^{th} shadow model. For all training data of the i^{th} shadow model, the prediction vector $\mathbf{Y} = f_{Shadow}^{i}(x)$ is generated and the record $(y, \mathbf{Y}, in)$ is added to the training dataset of the attack model where the prediction vector is $\mathbf{Y} = f_{Shadow}(x)$. Again, let, $D_{Shadow}^{Test^i}$ be the test dataset disjoint from the training dataset of the i^{th} shadow model. Then, $\forall (x, y) \in D_{Shadow}^{Test^i}$, the prediction vector $\mathbf{Y} = f_{Shadow}^{i}(x)$ is computed and the record $(y, \mathbf{Y}, out)$ is added to the training dataset of the attack model. Same procedure is applied to all the shadow models to get the training set, D_{Attack}^{Train} for the attack model. Shokri *et al.* [78] splits the dataset D_{Attack}^{Train} into c_{Target} partitions, each representing an independent class of the target model and train a separate model for each class that predicts the "in" or "out" status of x. The overall membership inference attack on machine learning models has been illustrated in Figure 3.2.

3.4 Conclusion

This chapter comprised of the details on membership inference attack on machine learning models. We discussed about different data generation techniques for the shadow model and also the data generation technique for the attack model. This

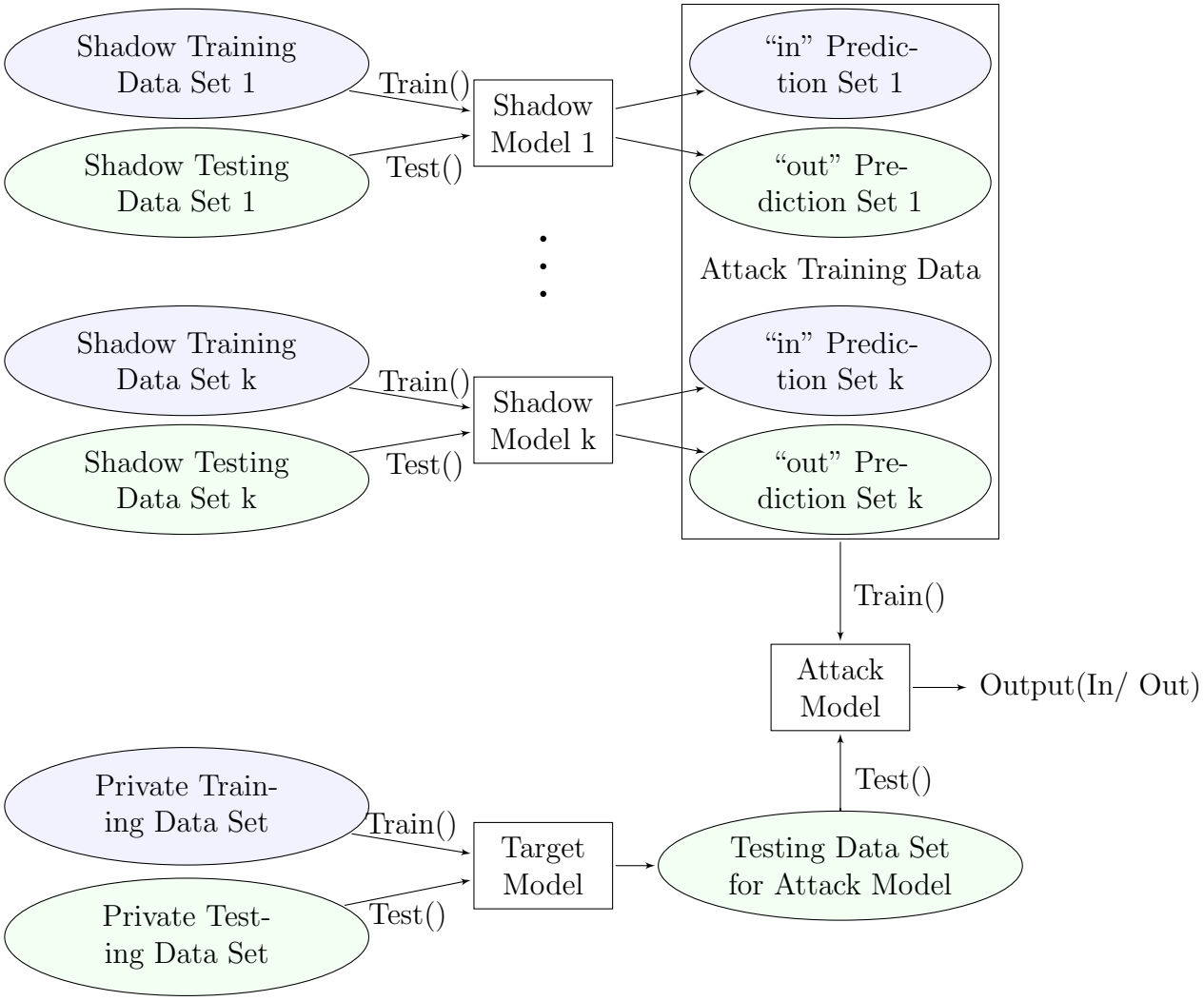

Figure 3.2: Membership Inference Attack on ML models [78]

chapter tells us about the process of attacking the machine learning model by an attacker to infer the members of the private training data.

Chapter 4

Analytical Results of Membership Inference Attack

This chapter discusses new combinations of parameters and settings, which were not explored in the literature to provide useful insights about the behavior of the membership inference attack. We conducted an empirical study to characterize the attack from different perspectives: number of shadow models, number of attack models, confidence thresholds, and different combinations of training algorithms for the different models included in the membership inference attack: the target model, shadow models and the attack model

4.1 Experimental Setup

We used MNIST dataset [42] for our experiment to evaluate the membership inference attack. This is a dataset of $70,000$ handwritten digits formatted as 32×32 images and normalized so that the digits are located at the center of the image. We performed our experiments on 2.6 GHz 6-Core Intel Core $i7$ processor with 32 GB 2667 MHz DDR4 memory in macOS Catalina version 10.15.5. We used $10,000$ randomly selected images for our target model of which $9,000$ images were used for

training the target model and the remaining $1,000$ images were used to test the target model.

- *Feed-forward Neural Network (FFNN) [50].* Its different parameters are detailed in Table 4.1.

Name	**Parameter**
Number of hidden layer(s)	1
Number of nodes in hidden layer	1000
Activation function for hidden layer	tanh
Activation function for output layer	softmax
Learning rate	0.0001
Number of epoch	1000
Batch size	1000
L-2 ratio	$1e^{-15}$

Table 4.1: Experimental Setup for FFNN

- *Softmax regression.* The learning rate was 0.0001 and the $L2$- regularization was $1e^{-15}$. Softmax regression is a generalization of logistic regression used for multiclass classification.

- *Convolutional Neural Network (CNN).* Its parameters are detailed in Table 4.2.

Name	**Parameter**
Input Shape	28×28
Number of Convolution Layer	2
Number of Max Pooling Layer	2
Strides (in each layer)	$(5, 5)$
Max Pooling (in each layer)	2×2
Hidden Layer Activation	tanh
Output Layer Activation	Softmax
Learning Rate	0.001
Batch Size	1000
Number of epoch	200

Table 4.2: Experimental Setup for CNN

- *Logistic regression.* We implemented it using the python library, Scikit-learn [68]. We used the "LBFGS" [7] solver of logistic regression with $L2$-regularization. The inverse of our regularization strength was 0.0001. We used the "ReLU" [59] as the activation function. All other meta parameters like the batch size, train-test data ratio, and the number of epochs remained the same to the ones used with FFNN and softmax regression.

The epoch and batch size is the same for FFNN, softmax regression, and logistic regression. We took $1,000$ data in a single batch and run $1,000$ epochs in a single experiment through out our work. For CNN we used 200 epochs whereas the batch size remains the same. We generated different number of shadow models for our different experiments. We explored the statistics-based data generation to generate the data for the shadow model training in most of the cases by sampling the value of each feature from its own marginal distribution. The attack models are very much effective in this case. We also explored the model-based data generation [78] by changing the confidence threshold ($conf_{min}$). The resulting attack gives variable attack accuracy depending on the threshold. We generated $10,000$ records for each shadow model of which 90% were used for training and 10% for testing.

4.2 Evaluation Metrics

There are different metrics for measuring the accuracy of our attack model. Some of them are:

1. **Classification Accuracy:** Measures the ratio of correct predictions to the total number of input samples [53].

2. **Logarithmic Loss:** Works by penalising the false classification [58].

3. **Confusion Matrix:** Describes the complete performance of the model as true positives, true negatives, false positives and false negatives [82].

4. **Area Under Curve:** This ranks the probability of a randomly chosen positive example higher than the negative example. The ranking depends on the sensitivity and specificity of the data [29].

5. **Mean Absolute Error:** The average of the difference between the original values and the predicted values [94].

6. **Mean Squared Error:** It takes the average of the square of the difference between the original values and the predicted values [90].

7. **Precision:** It is the ability of a model to return only relevant instances [65].

8. **Recall:** It is the ability of a model to identify all relevant instances [65].

The standard metrics for measuring the attack accuracy are the precision and recall. In the attack model, the precision represents what fraction of records inferred as members are indeed members of the training dataset.

$$precision = \frac{truepositives}{truepositives + falsepositives}$$

But recall represents what fraction of the members of the training dataset are correctly inferred as members by the attacker.

$$recall = \frac{truepositives}{truepositives + falsenegatives}$$

As recall focused on correctly inferred members of the training dataset, we considered it as the evaluation metrics in our experiments.

4.3 Analysis of Different Parameters on Membership Inference Attack

We analyzed the membership inference attack on machine learning models from different perspectives.

- Number of shadow models
- Type of training algorithms used to train the the attack model, the target model, and the shadow models
- Number of attack models
- Confidence score value

4.3.1 Number of Shadow Models

We used model-based synthesis to generate the data for the shadow models. We conducted the experiment to analyze the effect of the number of shadow models on our classification model (FFNN) and regression model (logistic regression).

4.3.1.1 Classification

We conducted two different experiments to analyze the effect of the number of shadow models on the membership inference attack. Shokri *et al.* [78] stated that the more the number of shadow models, the more accurate the attack model will be, but did not give empirical results to support this hypothesis. For the first experiment, we empirically tested the effect of the number of shadow models on the attack. As shown in Figure 4.1, when we increased the number of shadow models, more data was generated for the attack model which created more accurate and more precise attack model but the cost of the attack also increased. Attack accuracy is higher when more shadow models are used. With 15 shadow models our attack accuracy

become 95% while with using only 1 shadow model the attack accuracy was only 84%. We can notice that the accuracy change is not that much significant after a certain number of shadow models.

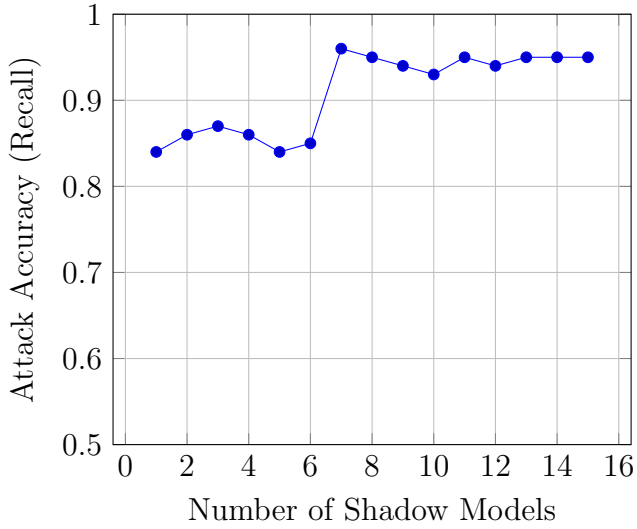

Figure 4.1: Effect of Number of Shadow Models on Classification

Shokri *et al.* [78] used the same machine learning technique to train the target model and shadow models. Shadow models can be trained using a machine learning technique that is different than the one used to train the target model. Additionally, two different machine learning techniques can be used together to train shadow models. We conducted experiments to analyze these settings and compared the results with Shokri *et al.*'s [78] method. Figure 4.2 shows the results of these experiments on the membership inference attack. In the "same" case, we used the FFNN to train the shadow models and the target model. In the "different" case, we trained the shadow models using a different algorithm than that used to train the target model. We used the FFNN to train the target model whereas we trained the shadow models using the softmax regression. In the "mixed" case, we used mixed types of training algorithm for the shadow models to generate the data for the attack model. We trained half of the shadow models using the FFNN while the other half were trained

using the softmax regression. It is to be mentioned that, we used FFNN to train the target model in all the above mentioned settings. We conducted these experiments by changing the number of shadow models. When increasing the number of shadow models, mixed types of shadow models gives better attack accuracy than the "same" and "different" cases. In the "mixed" case, diversified data was generated for the attack model which influenced better the attack model training. As a result, the attack model generated more attack accuracy. However, we can say that the "same" case is almost the best one with different numbers of shadow models.

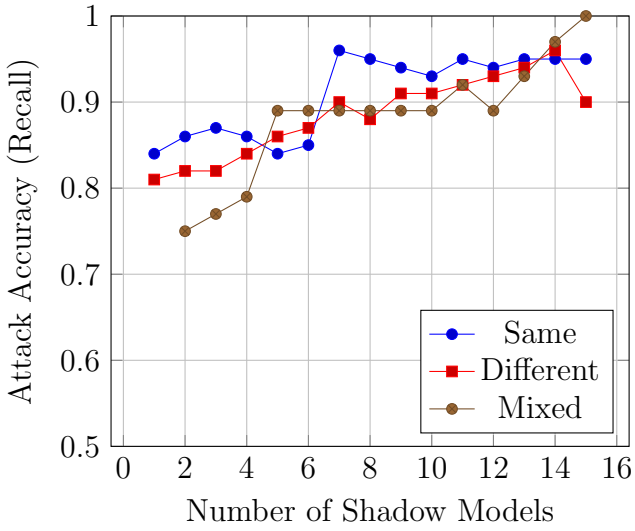

Figure 4.2: Types of Shadow Models Training in Contrast to the Number of Shadow Models

4.3.1.2 Regression

We also conducted an experiment to show the effect of number of shadow models on the logistic regression based target model. In this experiment, we trained all the three types of models (target model, shadow models and attack model) using logistic regression. Similar to Figure 4.1, Figure 4.3 proves the hypothesis of Shokri *et al.* [78]. The attack model inferred 91% of the members with 15 shadow models. However, we can notice that the attack recall is more for the FFNN-based target

model.

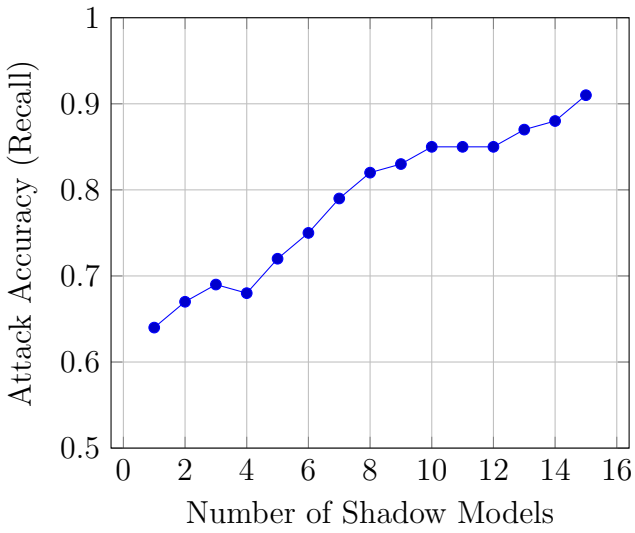

Figure 4.3: Effect of Number of Shadow Models for Logistic Regression Model

4.3.2 Effect of Training Algorithms

Machine learning models behave differently depending on the algorithm they are trained on.

4.3.2.1 Classification

In the membership inference attack, there are three different types of machine learning models namely the target model, the shadow models and the attack model (See Chapter 3 for details). We used two different algorithms for our classification model, FFNN and softmax regression. We followed the below criteria for training our three models.

- Same algorithm to train the target model and the shadow models
- Same algorithm to train the target model and the attack model
- Same algorithm to train the shadow models and the attack model

To consider the above mentioned facts, we made a combination of algorithms mentioned in Table 4.3 to show the impacts of training algorithms on membership inference attacks. We considered these different combinations with two different approaches:

- First approach is proposed by Shokri *et al.* [78] where the attack model comprised of multiple attack models each representing a particular class label and there are multiple shadow models (10 shadow models).
- Second method is proposed by Salem *et al.* [74] where there is a single attack model along with a single shadow model.

No.	**Target Model**	**Shadow Model**	**Attack Model**
1	FFNN	FFNN	FFNN
2	FFNN	FFNN	Softmax
3	FFNN	Softmax	Softmax
4	FFNN	Softmax	FFNN
5	Softmax	Softmax	Softmax
6	Softmax	Softmax	FFNN
7	Softmax	FFNN	FFNN
8	Softmax	FFNN	Softmax

Table 4.3: Combinations of Algorithms for Different Models

As shown in Figure 4.4, there is no generalization for the two approaches. However, We can see that Shokri *et al.*'s [78] approach is doing better than Salem *et al.*'s [74] approach.

The target model leaked less information about the members of the training data when the target model was trained using softmax regression than with neural network. In Figure 4.4, the accuracy of membership inference is more for combinations $2, 3$, and 4. If any attacker has the information about the training algorithm of the

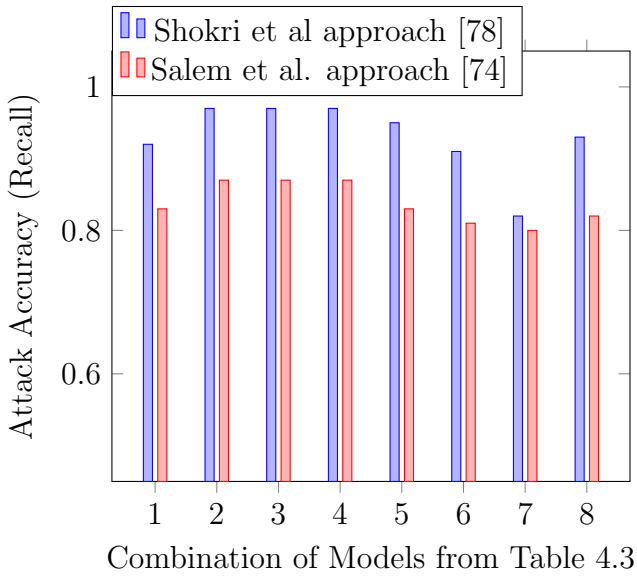

Figure 4.4: Effect of Type of Training Algorithms

target model, then this will increase the accuracy of the attack. If either of the training algorithm of the shadow models or the attack model are trained using the softmax regression, the attack accuracy becomes more as shown in Figure 4.4.

4.3.2.2 Regression

We also fixed the training algorithm for the shadow models and the target model to logistic regression and analyzed the situation if the attack model is trained with logistic regression one time and FFNN in the other. The attack accuracy in these settings is shown in Figure 4.5. For high number of shadow models, the attack model inferred more members when the training algorithm of the attack model is the same as that of the target model (logistic regresion).

4.3.3 Number of Attack Models

The number of attack models is another important factor in the membership inference attack. Shokhri *et al.* [78] used multiple attack models where each attack model represents a particular class label of the target model. We named this approach as

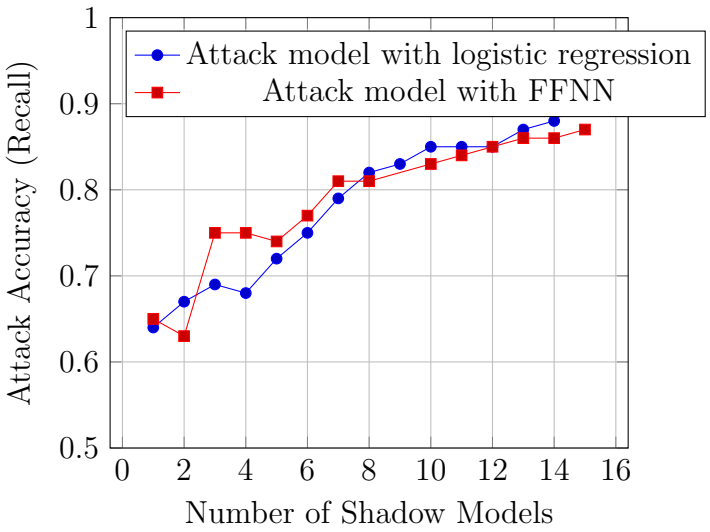

Figure 4.5: Effect of Type of Training Algorithm for the Target Model - Logistic Regression

class-specific attack models where if there are C_{Target} class labels in the target model, then there will be C_{Target} different attack models each representing a unique class of the target model. We conducted an experiment, which is called a class-independent attack model, where a single attack model is used in the membership inference attack. The main difference between this approach and Salem *et al.*'s approach [74] is having multiple shadow models (10 shadow models). Salem *et al.* [74] have a class-independent attack model and one shadow model whereas in our experiment we have a class-independent attack model with multiple shadow models. The benefit of the class-specific attack models is to train a specific model for each class. Likewise the previous experiments, we also studied the effect of the number of attack models on the membership inference attack from two different perspectives.

4.3.3.1 Classification

In this experiment, we used the combination of algorithms mentioned in Table 4.3 to evaluate the effects on the membership inference attack.

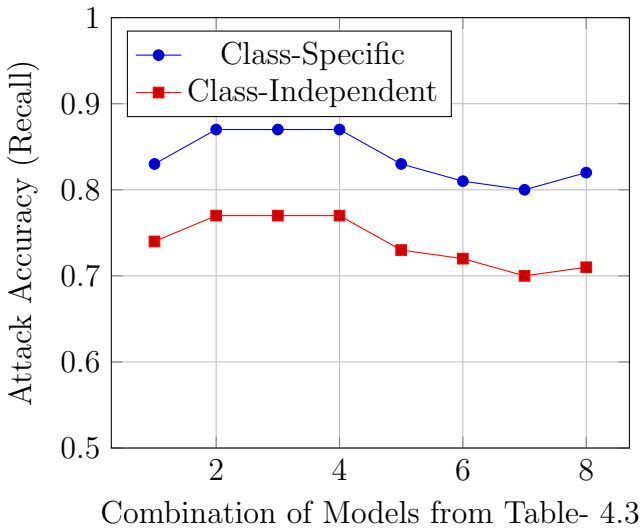

Figure 4.6: Effect of Number of Attack Models

From Figure 4.6, we can see that, the class-specific attack models give better attack accuracy than the class-independent attack model. As a class-specific attack model represents an individual class of the target data, it can infer more members from the target dataset than the class-independent attack model.

4.3.3.2 Regression

We also conducted a similar experiment using logistic regression by changing the number of attack models. The results of the experiment are illustrated in Figure 4.7. Class-specific attack models gives better attack accuracy in all cases whether the training algorithm of the attack model is logistic regression or FFNN. The target model and shadow models are trained using logistic regression in all cases.

4.3.4 Confidence Value

Confidence score is a threshold that determines what the lowest matching score acceptable to trigger an interaction is. If the matching score falls below the confidence score, it will not trigger an interaction. Previously, we generate our training data for the attack model based on the output of the shadow models and the testing data

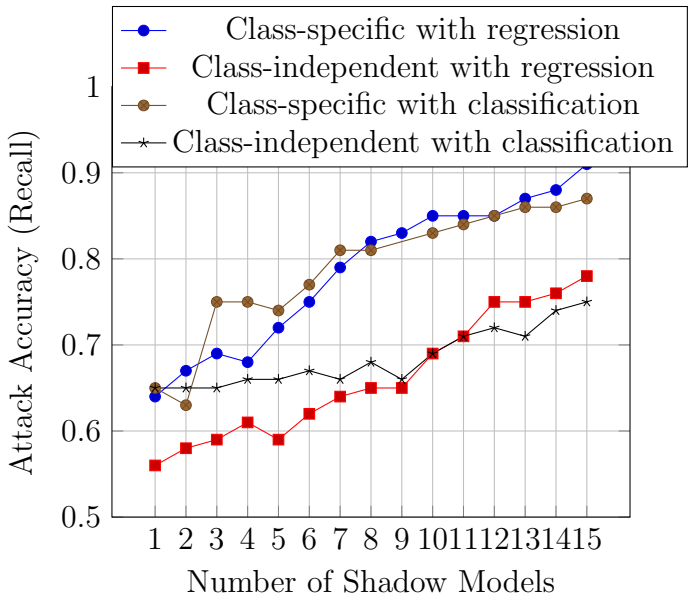

Figure 4.7: Effect of Type of Number of Attack Models on Logistic Regression

based on the output of the target model. We only considered the highest probability to classify a data, not any particular threshold. As a result of this, any class can be chosen with only $x\%$ probability where $x\%$ be the highest probability for that particular data.

In this experiment, we took the data for the training and testing of the attack model based on a particular threshold (70%). To get the highest confidence data, we removed any data classified with less than 70% probability whereas to get the low confidence data, we removed the data classified with more than 70% probability. Table 4.4 shows the effect of the confidence threshold on the membership inference attack. When removing data classified with more than 70%, the attack accuracy was 0 whereas when removing data classified with less than 70%, the attack accuracy was 0.75. We had 10 shadow models for this experiment. Additionally, all models were trained using the FFNN.

We performed another experiment by releasing data based on a threshold. We removed the data which were classified with probabilities more than a threshold $x\%$.

Type	**Training Accuracy of Target Model**	**Testing Accuracy of Target Model**	**Attack Accuracy (Recall)**
Low Confidence	1.0	0.9088	0
High Confidence	1.0	0.9088	0.75

Table 4.4: Effects of Confidence Score

We tried different values of threshold and performed the experiment (Table 4.5). For a threshold value of 80%, our attack model inferred about 66% of the data from the target model whereas for a threshold value of 40%, it is only 9%. We can conclude that the target model should define a low threshold where all data classified with probabilities more than this threshold will be removed when testing the attack model for membership. It is important to mention that Shokri *et al.* [78] has considered a threshold (0.2) just when generating the data for the shadow models.

Threshold	**Training Accuracy of Target Model**	**Testing Accuracy of Target Model**	**Attack Accuracy (Recall)**
0.2	1.0	0.9178	0
0.4	1.0	0.9178	0.09
0.6	1.0	0.9178	0.34
0.8	1.0	0.9178	0.66

Table 4.5: Attack Accuracy Based on Confidence Score

4.4 Conclusion

In this chapter, we analysed the membership inference attack on the basis of different parameters and settings. We first mentioned about our experimental setup for different training algorithms used for evaluating the attack followed by different evaluation metrics. Then we discussed about the experimental results based on our analysis.

Chapter 5

Mitigation Techniques

Successful membership inference attack helps in breaching the privacy of the members taking part in the model training. It is an urgent research problem to protect the privacy of the data taking part in training a ML model. The principle goal of our thesis is to protect the training data of the target model without compromising the utility of it from the membership inference attack. To fulfil our goal, in this research, we proposed and evaluated four different types of defence mechanisms against three types of target models. First of all, we proposed "Gaussian Noise Layer" to preserve the privacy of simple feed-forward neural network (FFNN) and convolutional neural network (CNN). We then proposed another method based on "Adversarial Noise Layer" [99] for convolutional neural network (CNN). We also proposed and evaluated "$L2-$regularizer" as a defence mechanism for a logistic regression model. Lastly, we explored the idea of "Exponential Mechanism" [49] as a mitigation technique for all the three types of training algorithms. To the best of our knowledge, we are the first to conduct membership inference attack on regression models till date. We discussed the details of our mitigation techniques along with the results in this chapter.

5.1 Gaussian Noise Layer

We proposed and implemented the "Gaussian Noise Layer (GNL)" as a privacy-preserving technique for a neural network based target model (FFNN and CNN). The statistical noise containing a Probability Density Function (PDF) which is equal to the normal distribution also known as the Gaussian distribution [103] is called Gaussian noise. The probability density function, PDF of a Gaussian Distribution for a random variable x is:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5.1}$$

where, μ is the mean and σ is the standard deviation.

5.1.1 Feed-forward Neural Network

A feed-forward neural network (FFNN) [50] is a type of neural networks where data passes through different input nodes until the output node in a forward direction. This type of neural network is also known as front propagated wave which is achieved by using a classifying activation function. The data moves in one direction only and no back propagation happens in this neural network. In FFNN, the sum of the products of the inputs and their weights are calculated and fed to the next layer. Based on the activation function used in that particular layer, the neuron will be activated or remain inactive.

A simple neural network consists of three layers: an input layer, hidden layer and output layer. We added a new layer after the output layer and also after the hidden layer named a Gaussian noise layer in our mitigation technique for a FFNN-based target model to regularize the target model.

We used Lasagne [12] to implement our target model using FFNN, which is a light weight Python library to build and train feedforward neural networks in Theano. Lasagne supports feed-forward networks, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) including Long Short-Term Memory (LSTM), and any combination thereof [12].

We used a simple feed-forward neural network with an input layer, one hidden layer, and one output layer. For ensuring the privacy of the target model, we used a noise layer in the network. The *lasagne.layers* module provides various classes representing the layers of a neural network which are subclasses of the *"lasagne.layers.Layer"* base class [12]. The *"InputLayer"* represents the input of the network while the *"HiddenLayer"* takes its input from the output of the *"InputLayer"* and the *"OutputLayer"* takes its input from the output of the *"HiddenLayer"*. We depend on the *"GaussianNoiseLayer"* of Lasagne as a noise layer in our mitigation technique. We added the noise layer [38] in two different positions which acts as an adversarial noise layer to regularize the neural network and help in preserving privacy by protecting the target model against membership inference attacks. The class used in our experiment is *lasagne.layers.GaussianNoiseLayer(incoming, sigma=0.1,**kwargs)*. This adds zero-mean Gaussian noise of a given standard deviation (*"sigma"*) to the input [38]. The *"incoming"* represents the layer instance feeding into this layer. In our experiments, it as either *"OutputLayer"* or *"HiddenLayer"*. The *"sigma"* represents the standard deviation of the added Gaussian noise termed as the noise level in our work. The argument *deterministic* is set to *"false"* during training and the same to *"true"* during testing. As a result, the noise layer acts as a regularizer during training and it does not have any effect during testing.

Figure 5.1 represents the training process of our privacy-preserved target model. We have used two different approaches by adding the noise layer after the output

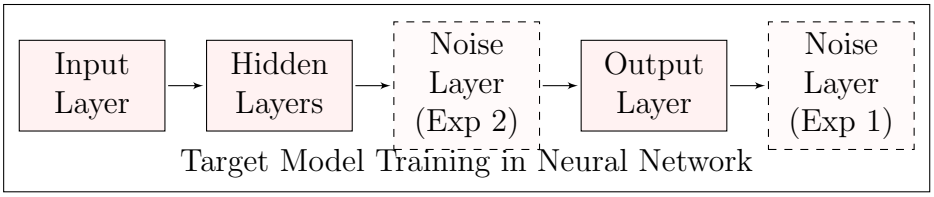

Figure 5.1: GNL-based Training of the Target Model

layer as well as by adding it after the hidden layer.

As we mentioned earlier, we used a simple network consisting of one input layer, one hidden layer and one output layer with the experimental settings detailed in Section 4.1. We performed two different experiments with GNL for the privacy of the target model (Figure 5.1). In our first experiment, we added GNL after the output layer where we used the output from the output layer as an input of GNL and the output from the GNL as the final output of the model. In our second experiment, we added GNL after the hidden layer where the GNL takes the output of the hidden layer as the input and the output of the GNL was used as an input for the output layer.

	GNL after output layer Experiment 1		**GNL after hidden layer** Experiment 2	
Noise Level	**Training Accuracy**	**Testing Accuracy**	**Training Accuracy**	**Testing Accuracy**
0	1	0.9137	1	0.9137
0.3	0.9975	0.934	1	0.9198
0.5	0.9917	0.9352	1	0.9192
0.8	0.9823	0.9357	1	0.9435
1.0	0.9758	0.9308	1	0.9437
1.2	0.9662	0.9247	1	0.942
1.5	0.9623	0.9187	1	0.941

Table 5.1: Accuracy of the Target Model - FFNN

The experimental results showed the effectiveness of GNL as a privacy preserving technique against the membership inference attack. We added GNL with different

sigma values (noise levels) in our target model. Table 5.1 shows the accuracy of the target model after adding different amount of noise to the target model trained using feed-forward neural networks for both experiments. The training accuracy of the target model is the highest for neural network with 0 noise level (no privacy). With the increase of noise level, the training accuracy of the target model decreases slightly and the testing accuracy generally increases. Additionally, when adding GNL after the output layer the accuracy of the target model with noise (level 1.5:$(0.9623, 0.9187)$) is almost the same to the accuracy of the target model without noise (level 0:$(1, 0.9137)$).

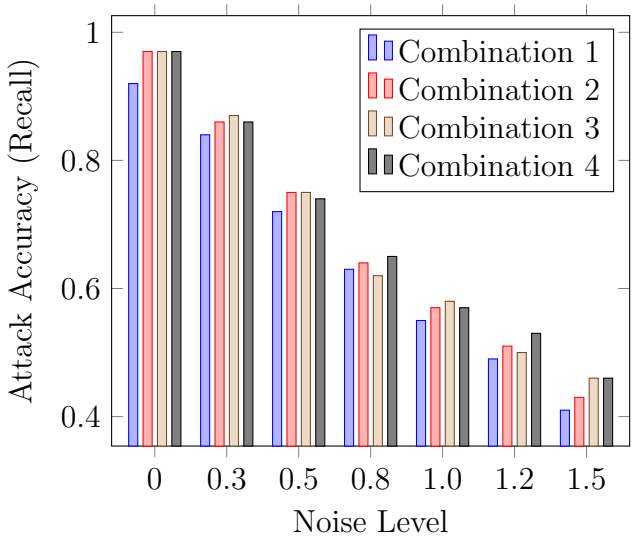

Figure 5.2: GNL after Output Layer

The accuracy of the attack, for different noise levels and different combinations (Table 4.3) when adding the GNL after the output layer and after the hidden layer of the FFNN are shown in Figure 5.2 and Figure 5.3, respectively. The privacy of the target model is increased while increasing the noise. With the addition of 1.5 noise, the data leakage reduced from 97% to 46%. If we add more noise to the target model, the data leakage will decrease more, but at the same time it will affect the effectiveness of the target model. The Gaussian noise layer is more effective when

added after the output layer of the neural network than that of after the hidden layer. We also tried to add the noise layer after the input layer but the results were not promising.

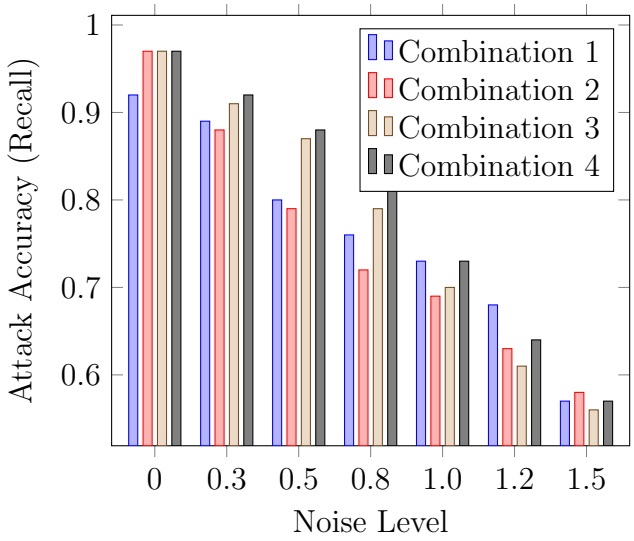

Figure 5.3: GNL after Hidden Layer

5.1.2 Convolutional Neural Network

We also explored the use of Gaussian noise layer of lasagne [12] with convolutional neural network (the experimental settings are discussed in Section 4.1) to preserve the privacy of the target model. Table 5.2 shows the accuracy of the target model for different noise levels. The training accuracy of the target model reduced from 0.9433 to 0.752 due to addition of 0.4 noise whereas the testing accuracy reduced from 0.9301 to 0.7522 for the same noise. It is to be mentioned that, we added the "Gaussian Noise Layer" of "Lasagne" after the output layer of the target model trained with CNN.

Figure 5.4 shows the accuracy of the attack model for different amount of noise in the target model. With the increase of noise, the accuracy of the attack decreases. With 0.4 amount of noise, the attack accuracy decreases from 77% to 56% but the

Noise Level	Training Accuracy	Testing Accuracy
0	0.9433	0.9301
0.1	0.9143	0.9064
0.15	0.9014	0.8994
0.2	0.8851	0.8757
0.25	0.858	0.8452
0.3	0.869	0.8654
0.35	0.8404	0.8368
0.4	0.752	0.7522

Table 5.2: Accuracy of the Target Model - CNN with GNL

accuracy of the target model also decreases much (See Table 5.2)

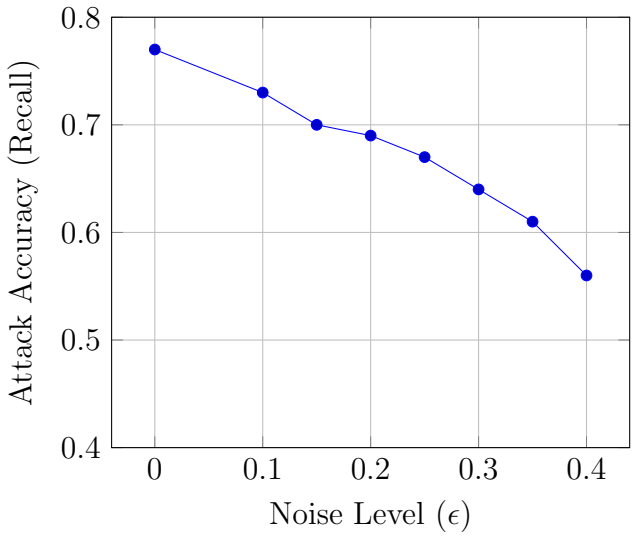

Figure 5.4: Privacy of the CNN-based Target Model with GNL

5.2 Adversarial Noise Layer

Although being powerful, Convolutional Neural Network (CNN) can face overfitting due to the excessive amount of parameters. You et al. [99] introduced a novel regularization method named as Adversarial Noise Layer (ANL) to overcome the overfitting problem of CNN. In this method, You et al. [99] added a carefully crafted noise to the intermediate layer activation of the CNN to improve the generalization

ability of the model. Noise is only added during the training process of the model and removed after the training ends.

Deep convolutional neural network architecture extracts the vision features one layer after another [101] which is the main concept to reduce the entropy loss of this type of model. Let us suppose that, h_t be the output of the t^{th} layer of the network and η_t be the adversarial noise related to h_t. The adversarial noise η_t is added to the h_t during training the model using Equation. 5.2 and $\bar{h}_t$ is passed to the next layer during training.

$$\bar{h}_t = h_t + \eta_t \tag{5.2}$$

The noise layer is removed after training the model and $\bar{h}_t$ is set to be h_t in the testing phase which removes the extra computation in the testing phase. The adversarial noise η_t is calculated on the basis of the gradient of h_t in equation 5.5.

$$r = Clip_{<0,\epsilon>}\{N(\epsilon/2, (\epsilon/2)^2)\} \tag{5.3}$$

$$g_t = \bigtriangledown_{h_t} J(x, y; \theta, \eta)|_{\eta=0} \tag{5.4}$$

$$\eta_t = rs(h_t)g_t/||g_t||_\infty \tag{5.5}$$

In the above equations, $Clip_{<(0,\epsilon)>}(A)$ denotes element wise clipping of A ($A = N(\epsilon/2, (\epsilon/2)^2)$) within range $[0, \epsilon]$, $J(x, y; \theta)$ is the cost function of the model, $s(h_t)$ is the standard deviation of h_t, $N(\epsilon/2, (\epsilon/2)^2)$ is the Gaussian-distribution where $\epsilon/2$ is the mean and $(\epsilon/2)^2$ is the standard deviation. The magnitude of the noise is controlled by a random scalar r and is multiplied by $s(h_t)$ to get a wide range of activation which helps in tolerating large perturbation.

Training with ANL follows a two-rounds-training strategy (illustrated in Figure 5.5 and Figure 5.6) where the adversarial noise is calculated after the first round and

the network is updated after the second round. For this, ANL requires an additional forward and backward propagation.

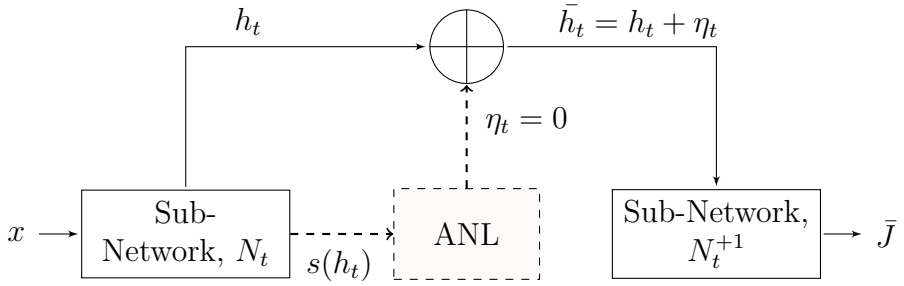

Figure 5.5: First Round of ANL

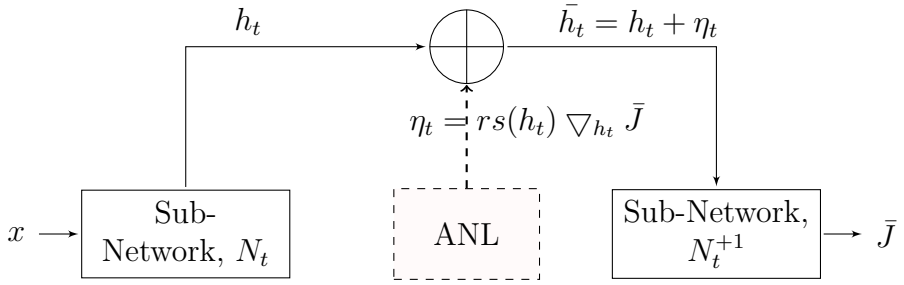

Figure 5.6: Second Round of ANL

ANL calculates the $s(h_t)$ in the forward phase of the first round and $\bigtriangledown_{h_t}\bar{J}$ in backward phase of the same round (Figure 5.5). ANL then generates the noise, η_t using $s(h_t)$ and $\bigtriangledown_{h_t}\bar{J}$ from the first round to update the network parameters in the second round during back-propagation (Figure 5.6). Algorithm 2 [99] shows the training process using ANL.

We studied the use of adversarial noise layer [99] with convolutional neural network trained target model to protect the membership inference of the target model. We have used the below architecture for our ANL-based privacy mechanism:

$$CONV \rightarrow ANL \rightarrow RELU \rightarrow POOL \rightarrow CONV \rightarrow ANL \rightarrow RELU \rightarrow POOL$$

Algorithm 2 Training with ANL [99]

procedure REQUIRE($\epsilon > 0$)
  while training complete **do**
    Sample a batch $\{X, Y\}$ from training data.
    Calculate $J(X, Y; \theta, 0)$ and standard deviation in forward-propagation.
    Calculate gradients in backward-propagation.
    for $t = 1$ to L **do**
      Update η_t ▷ Using equ. 5.5
    end for
    Get $J(X, Y; \theta, \eta)$ in second forward-propagation
    calculate the gradients in second backward-propagation
    Update network with $\theta = \theta - \lambda \bigtriangledown_\theta J(x, y; \theta, \eta)$
  end while
end procedure

We studied the accuracy of the target model for different amount of noise (ϵ) along with the attack accuracy to check whether our method maintains the trade-off between privacy and utility. Table 5.3 shows the accuracy of the target model for changing the noise level from 0 to 0.4. The training accuracy of the target model changed from 0.9655 to 0.9014 whereas the testing accuracy has been reduced from 0.9545 to 0.8994.

Noise Level	**Training Accuracy**	**Testing Accuracy**
0	0.9655	0.9545
0.1	0.9621	0.952
0.15	0.9552	0.9423
0.2	0.9492	0.9343
0.25	0.9433	0.9301
0.3	0.9308	0.9214
0.35	0.9143	0.9064
0.4	0.9014	0.8994

Table 5.3: Accuracy of the Target Model - CNN with ANL

Figure 5.7 illustrates the attack accuracy for different amount of noise in the target model. The attack accuracy reduced from 0.77 to 0.57 for adding 0.4 amount of noise in the target model. The change of noise in the target model maintaied

a trade off between the utility and privacy of the target model trained with CNN using ANL. Adding more noise can further reduce the membership inference but it will also reduce the utility of the target model. As a result, the trade off will not be maintained in the target model.

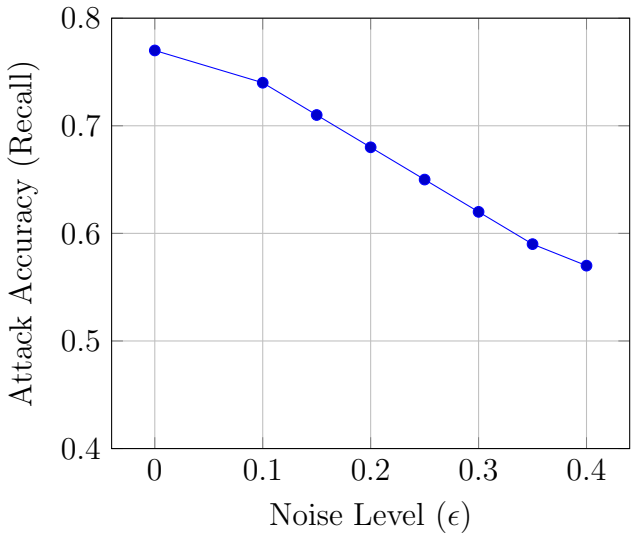

Figure 5.7: Privacy with ANL

We studied two methods (ANL and GNL) to ensure the privacy of the target model trained with CNN. From the illustrated results shown in Figure 5.4 and Figure 5.7, the attack accuracy of the membership inference is almost the same for the two methods but the utility is better maintained for CNN with ANL than CNN with GNL.

5.3 L2-Regularization

Regularization is one of the most important technique to overcome the overfitting problem of machine learning models. It helps in maintaining the trade off between the complexity and flexibility of ML models. Regularization is a modification in learning algorithms that reduces the generalization error but not the training error [25]. The

objective function (negative of the log likelihood function) is minimized to calculate the regression coefficients of a logistic regression.

$$\bar{\beta} = min_{\beta}[-LL(\beta; y, X)] \tag{5.6}$$

Here, LL is the log likelihood function, β is the coefficients, y is the dependent variable (class), and X is the independent variable (features). A regularization term, $R(\beta)$ is added to the objective function $\bar{\beta}$ multiplied by a parameter $\lambda \in R_{+}$ to penalize the high coefficients.

$$\bar{\beta} = min_{\beta}[-LL(\beta; y, X) + \lambda R(\beta)] \tag{5.7}$$

The penalization is added to the high coefficients of logistic regression to overcome assigning any particular class for a single feature [39]. Different types of regularization can be added to penalize the high coefficients. Among them, we studied the $L2-$regularization to generalize our target model and help preventing membership inference attack for the regression model.

$L2-$regularization (also known as ridge regression) adds the squared magnitude of coefficients to the loss function as a penalty term to help generalizing a machine learning model. $L2-$regularization can be defined as

$$R(\beta) = 1/2 \sum_{i=0}^{n} \beta_i^2 \tag{5.8}$$

which is the sum of the squared of the coefficients multiplied by $1/2$. Therefore, the overall cost function becomes,

$$\bar{\beta} = min_{\beta}[-LL(\beta; y, X) + \lambda(1/2 \sum_{i=0}^{n} \beta_i^2)] \tag{5.9}$$

Here, λ is the only meta-parameter to control the regularization which gives smaller coefficients for higher values. Too high value for λ might lead the model to underfitting problem.

We investigated how $L2-$regularization can protect the target model against membership inference attack when trained with logistic regression. "Logit" [11] function has been used to convert the continuous value of a regression model into a class. It is important to mention that, Truex *et al.* [87] studied the impact of noisy target data and shadow data size on the attack accuracy with logistic regression models. Here, we studied the effect of the target model's $L2-$regularization parameter against membership inference attacks.

$L2-$regularization helps in regularizing a model and also helps to overcome the overfitting problem of a machine learning model. We decreased the $L2-$regularization parameter to protect the target model against the membership inference attack. Table 5.4 shows how the accuracy of our target model changes with the regularization. We trained our target model and the shadow models with logistic regression but changed the way of training the attack model. We used both logistic regression and feed-forward neural network (FFNN) to train our attack model in two different experiments. When changing the regularization parameter from 0.05 to 0.000001, our target model training accuracy decreases from 100% to 94%. Meanwhile, the testing accuracy increases from 86% to 91%. This helps in reducing the overfitting of the target model and thus helps in protecting the target model.

Figure 5.8, illustrates how the attack accuracy reduces with the use of regularization in the target model. As we strengthen the regularization in our target model, the target model reduces its overfitting and becomes less prone to membership inference attacks. Our attack model with logistic regression algorithm can infer about 100% of

Index	Regularization	Training Accuracy	Testing Accuracy
1	0.05	1.0	0.86
2	0.01	1.0	0.8635
3	0.005	1.0	0.865
4	0.001	1.0	0.87
5	0.0005	0.99	0.87
6	0.0001	0.99	0.88
7	0.00005	0.98	0.90
8	0.00001	0.96	0.91
9	0.000005	0.95	0.91
10	0.000001	0.94	0.91

Table 5.4: Accuracy of the Target Model - Logistic Regression

the members from the target model with 0.05 regularization and it is the same when we train the attack model using the FFNN. In the first case (logistic regression), all models are trained using logistic regression whereas in the second case (FFNN) the attack model is trained using the FFNN whereas the shadow models and the target model are trained using the logistic regression algorithm. With the change of regularization in the target model, the attack accuracy of the attack model trained either with the logistic regression or FFNN algorithm changes rapidly. With only 0.0001 regularization, the attack accuracy reduced to 84% for the attack model trained with regression and 92% for or the attack model trained with FFNN. It is further reduced to 54% for regression and 70% for FFNN with 0.000001 regularization.

5.4 Exponential Mechanism

Exponential Mechanism [49] is the fundamental tool of differential privacy that naturally fits with estimation techniques of machine learning as well as statistics [3]. Assigning higher density values to the regions with high utility is the main idea behind this mechanism. To preserve the differential privacy, exponential mechanism chooses an optimal output close to the utility function [56]. A probability distribution over the output range is induced which sampled the output based on the sensitivity

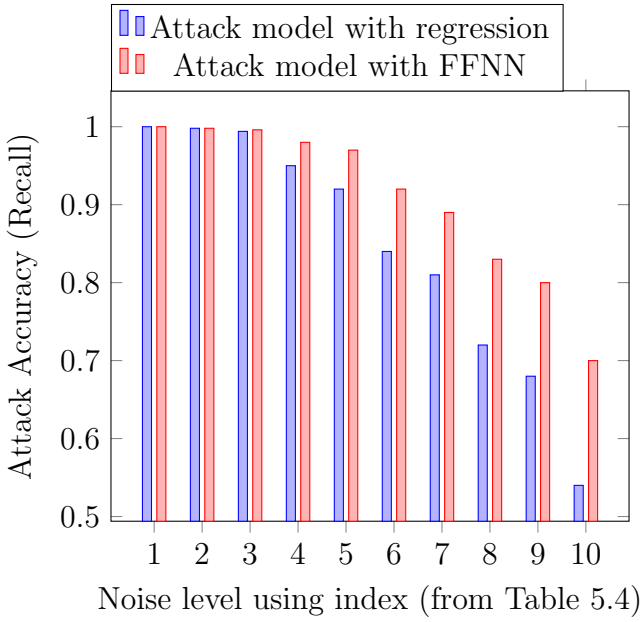

Figure 5.8: Privacy of the Target Model - Logistic Regression

of the utility function in this mechanism. The output with a higher probability is exponentially more likely to be chosen by this exponential mechanism [56].

We explored the exponential mechanism for protecting the data taking part in the training of the target model. The target model output the probability score vector for each data point of the training dataset. Implementation of the exponential mechanism is to have the interval $[0, 1]$ (as the sum of the probability score vector is 1), partitioned into segments. We sample a random number uniformly in the range $[0, 1]$ and the partition in which the random number falls determines the winner class. Algorithm 3 details this mitigation technique based on the idea of the exponential mechanism.

We explored this idea based on the exponential mechanisms for three types of training algorithms for the target model. We detailed our experimental results as below:

Algorithm 3 Exponential Mechanism based Mitigation Technique

```
procedure REQUIRE(probability vector of each data points)
    for datapoint = 1 to n do
        Chose a random number r within [0, 1]
        Make a range c_i with the probabilities
        if r in range c_i then
            Return c_i
        end if
    end for
end procedure
```

5.4.1 Feed-forward Neural Network

Our first target model training algorithm was the feed-forward neural network (FFNN). We used the idea related to exponential mechanism with the FFNN based target model. Table 5.5 showed the experimental result of our method with FFNN based target model where the shadow models and the attack model used the same algorithm (FFNN) to train themselves.

Method	**Target Model Training Accuracy**	**Target Model Testing Accuracy**	**Attack Accuracy**
Without Privacy	1.00	0.9137	0.92
With Privacy	1.00	0.945	0.53

Table 5.5: Exponential Mechanism on Feed-Forward Neural Network

The attack accuracy of our privacy preserving mechanism reduces the attack accuracy from 92% to 53% for a FFNN based target model. The training accuracy of the target model remained the same (100%) for the target model trained with privacy and without privacy, but the testing accuracy increases for our privacy preserved target model from 91.37% to 94.5%.

5.4.2 Convolutional Neural Network

We also explored the idea related to the exponential mechanism with our target model trained with convolutional neural network (CNN). Table 5.6 illustrates the effectiveness of the technique in protecting the data leakage from membership inference attack. The shadow models were trained using CNN whereas the attack model was trained with softmax regression in this experiment.

Method	**Target Model Training Accuracy**	**Target Model Testing Accuracy**	**Attack Accuracy**
Without Privacy	0.9655	0.9545	0.77
With Privacy	0.963	0.9412	0.54

Table 5.6: Exponential Mechanism on Convolutional Neural Network

In case of CNN, this technique reduces the attack accuracy from 77% to 54%. The training and testing accuracy of the target model remains almost the same. With a loss of minimal utility of the target model, we got about 23% less successful attack by the attacker in inferring the members taking part in the training of the target model.

5.4.3 Logistic Regression

Our final training algorithm for the target model was the logistic regression. We used the idea related to the exponential mechanism for protecting the training data of the target model trained with logistic regression. In this experiment, the shadow models and the attack model were trained using logistic regression. The results of this experiment are illustrated in Table 5.7

This technique reduces the membership inference attack for the target model trained with logistic regression. The attack accuracy decreases from 85% to 51% for logistic regression-based target model. The testing accuracy of the target model

Method	Target Model Training Accuracy	Target Model Testing Accuracy	Attack Accuracy
Without Privacy	1.00	0.86	0.85
With Privacy	0.95	0.942	0.51

Table 5.7: Exponential Mechanism on Logistic Regression

increases from 86% to 94.2% though the training accuracy reduces about 5% with the use of it.

5.5 Comparative Analysis

We compared the existing mitigation strategies against membership inference attacks on machine learning models (summarized in Section 2.3). The comparison has been conducted using the MNIST dataset. We used the same number of records for all of the existing techniques. We used the settings of each mitigation technique specified by authors as they got their best results using these settings.

Mitigation Strategy	Author	Training Algorithm for Target Model	Type of Target Model	Training Accuracy of Target Model	Testing Accuracy of Target Model	Accuracy (Recall) of Attack Model
Regularization	Shokri et al. [78]	FFNN	Classification	0.9635	0.9146	0.77
Min-Max Game	Nasr et al. [61]	FFNN	Classification	0.9564	0.9366	0.66
Differential Privacy	Jia et al. [37]	FFNN	Classification	0.97	0.94	0.63
GNL (Lasagne)	Our experiment	FFNN	Classification	0.9623	0.9187	0.46
Exponential Mechanism	Our experiment	FFNN	Classification	1.00	0.945	0.53

Table 5.8: Mitigation Strategy Comparison with Feed-forward Neural Network-based Target Model

Our first comparison is for the feed-forward neural network (FFNN)-based target model which is shown in Table 5.8. The min-max game theory proposed by Nasr *et al.* [61] gives better utility but differential privacy by Jia *et al.* [37] gives better utility and privacy trade off. If we consider about some utility of the target model, our proposed method of "Gaussian Noise Layer" gives better privacy guarantee than any other method on FFNN-based target model. It reduces the accuracy of the target model to 46% which is less than the random guess. Exponential mechanism also gives better privacy than other systems if we consider a little about the utility of the target model.

Mitigation Strategy	**Author**	**Training Algorithm for Target Model**	**Type of Target Model**	**Training Accuracy of Target Model**	**Testing Accuracy of Target Model**	**Accuracy (Recall) of Attack Model**
Dropout	Salem et al. [74]	CNN	Classification	0.9912	0.9841	0.62
Model Stacking	Salem et al. [74]	CNN	Classification	0.9912	0.9841	0.63
GNL (Lasagne)	Our experiment	CNN	Classification	0.752	0.7522	0.56
Adversarial Noise Layer	Our experiment	CNN	Classification	0.9014	0.8994	0.57
Exponential Mechanism	Our experiment	CNN	Classification	0.963	0.9412	0.54

Table 5.9: Mitigation Strategy Comparison with Convolutional Neural Network-based Target Model

Our next comparison is for the convolutional neural network (CNN)-based target model. In Table 5.9, model staking and dropout are the best mitigation strategy for the CNN-based target model in terms of utility-privacy trade off. But our proposed method of exponential mechanism is best if we consider about the utility of the target model. The accuracy of the attack model using exponential mechanism is 54% where

the same for dropout and model stacking are 62% and 63% respectively. Gaussian noise layer is the worst among all the mitigation techniques used with CNN-based target model against the membership inference attack in terms of utility-privacy trade off.

Mitigation Strategy	**Author**	**Training Algorithm for Target Model**	**Type of Target Model**	**Training Accuracy of Target Model**	**Testing Accuracy of Target Model**	**Accuracy (Recall) of Attack Model**
L2-regularization	Our experiment	Logistic regression	Regression	0.94	0.91	0.54
Exponential Mechanism	Our experiment	Logistic regression	Regression	0.95	0.942	0.51

Table 5.10: Mitigation Strategy Comparison for Logistic Regression-based Target Model

We proposed and evaluated two different mitigation techniques for logistic regression-based target model against the membership inference attack. To the best of our knowledge, we are the first to evaluate the mitigation strategy for the logistic regression-based target model. Between the two methods, exponential mechanism gives the better utility-privacy trade off against the membership inference attack for logistic regression-based target model.

Table 5.11 shows the whole overall comparison of the mitigation strategy evaluated against the membership inference attack on machine learning models. Most of the mitigation strategy are specific for a particular training algorithm of the target model. Only the exponential mechanism is used for all the three types of algorithms used in evaluating the membership inference attack in this thesis. If we consider about the utility of the target model ie. for more sensitive data we can use the exponential mechanism to protect our data from membership inference attack. But for less

Mitigation Strategy	**Author**	**Training Algorithm for Target Model**	**Type of Target Model**	**Training Accuracy of Target Model**	**Testing Accuracy of Target Model**	**Accuracy (Recall) of Attack Model**
Regularization	Shokri et al. [78]	FFNN	Classification	0.9635	0.9146	0.77
Min-Max Game	Nasr et al. [61]	FFNN	Classification	0.9564	0.9366	0.66
Differential Privacy	Jia et al [37]	FFNN	Classification	0.97	0.94	0.63
GNL (Lasagne)	Our experiment	FFNN	Classification	0.9623	0.9187	0.46
Dropout	Salem et al. [74]	CNN	Classification	0.9912	0.9841	0.62
Model Stacking	Salem et al. [74]	CNN	Classification	0.9912	0.9841	0.63
GNL (Lasagne)	Our experiment	CNN	Classification	0.752	0.7522	0.56
Adversarial Noise Layer	Our experiment	CNN	Classification	0.9014	0.8994	0.57
L2-regularization	Our experiment	Logistic regression	Regression	0.94	0.91	0.54
Exponential Mechanism	Our experiment	FFNN	Classification	1.00	0.945	0.53
Exponential Mechanism	Our experiment	CNN	Classification	0.963	0.9412	0.54
Exponential Mechanism	Our experiment	Logistic regression	Regression	0.95	0.942	0.51

Table 5.11: Mitigation Strategy Comparison

sensitive data, where utility is the major concern, we can use any mitigation strategy depending on the algorithm used to train the model.

5.6 Conclusion

We proposed and evaluated four different types of mitigation techniques against the membership inference attack in this chapter. We explored the "Gaussian Noise Layer" of "Lasagne" to protect both Feed-Forward Neural Network (FFNN)-based target models and Convolutional Neural Network (CNN)-based target models. We also used "Adversarial Noise Layer" to protect CNN-based target models. We were the first to implement "$L2-$Regularization" as a mitigation strategy for logistic regression-based model till date. The last mitigation technique we proposed against the membership inference attack is the exponential mechanism. We implemented this mechanism with all the three algorithms mentioned to mitigate the attack. Last of all, we compared our proposed mitigation techniques with the existing ones in this chapter.

Chapter 6

Conclusion

Machine learning models are vulnerable to various types of security and privacy breaches due to its popularity. An adversary tries to disclose the members of the training data by conducting the membership inference attack. In this work, we investigated different parameters and settings to provide useful insights about the behavior of the membership inference attack and the different factors for a successful attack.

We investigated three different mitigation techniques to defend the membership inference attack for four different types of training algorithms: Gaussian Noise Layer (GNL) for Feed-Forward Neural Network (FFNN) and Convolutional Neural Network (CNN), Adversarial Noise Layer (ANL) for CNN, $L2-$regularization for logistic regression and last of all exponential mechanism for all the three algorithms mentioned (i.e. FFNN, CNN, and Logistic Regression). We also compared all the existing mitigation techniques with our techniques. Our results showed a compromise between the privacy and utility of the model. It is to be mentioned that, we are the first to investigate the membership inference attack on logistic regression models and propose mitigation techniques against this attack in this setting.

We conducted the membership inference attack on two types of machine learning models- classification and regression. There are other types of models namely ensemble learning. This might be a good research scope to investigate the attack on this type of model. The mitigation techniques presented in our work are applied for a target model trained with a specific training algorithm. It would be a good future research scope to find a mitigation technique which will mitigate the membership inference attack for any type of training algorithm of the target model without compromising the utility of the model.

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang, *Deep learning with differential privacy*, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 308–318.

[2] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici, *Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers*, International Journal of Security and Networks **10** (2015), no. 3, 137–150.

[3] Jordan Awan, Ana Kenney, Matthew Reimherr, and Aleksandra Slavković, *Benefits and pitfalls of the exponential mechanism with applications to hilbert spaces and functional PCA*, 2019.

[4] Michael Backes, Pascal Berrang, Mathias Humbert, and Praveen Manoharan, *Membership privacy in MicroRNA-based studies*, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 319–330.

[5] Raef Bassily, Adam Smith, and Abhradeep Thakurta, *Private empirical risk minimization: Efficient algorithms and tight error bounds*, 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, IEEE, 2014, pp. 464–473.

[6] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser, *Machine learning classification over encrypted data.*, NDSS, vol. 4324, 2015, p. 4325.

[7] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM Journal on scientific computing **16** (1995), no. 5, 1190–1208.

[8] Nicholas Carlini and David Wagner, *Towards evaluating the robustness of neural networks*, 2017 ieee symposium on security and privacy (sp), IEEE, 2017, pp. 39–57.

[9] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate, *Differentially private empirical risk minimization*, Journal of Machine Learning Research **12** (2011), no. 29, 1069–1109.

[10] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz, *GAN-Leaks: A taxonomy of membership inference attacks against generative models*, Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (New York, NY, USA), CCS '20, Association for Computing Machinery, 2020, p. 343–362.

[11] James S Cramer, *The origins and development of the logit model*, Logit models from economics and other fields **2003** (2003), 1–19.

[12] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gábor Takács, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degrave, *Lasagne: First release.*, August 2015.

[13] Irit Dinur and Kobbi Nissim, *Revealing information while preserving privacy*, Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (New York, NY, USA), PODS '03, Association for Computing Machinery, 2003, p. 202–210.

[14] Pedro Domingos, *A few useful things to know about machine learning*, Commun. ACM **55** (2012), no. 10, 78–87.

[15] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville, *Adversarially learned inference*, 2017.

[16] Cynthia Dwork, *Differential Privacy: A survey of results*, International conference on theory and applications of models of computation, Springer, 2008, pp. 1–19.

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith, *Calibrating noise to sensitivity in private data analysis*, Theory of cryptography conference, Springer, 2006, pp. 265–284.

[18] Cynthia Dwork and Aaron Roth, *The algorithmic foundations of differential privacy*, Found. Trends Theor. Comput. Sci. **9** (2014), no. 3–4, 211–407.

[19] Cynthia Dwork, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan, *Robust traceability from trace amounts*, 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, IEEE, 2015, pp. 650–669.

[20] Bálint Fazekas and Attila Kiss, *Statistical data generation using sample data*, European Conference on Advances in Databases and Information Systems, Springer, 2018, pp. 29–36.

[21] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart, *Model inversion attacks that exploit confidence information and basic countermeasures*, Proceedings

of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 1322–1333.

[22] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart, *Privacy in Pharmacogenetics: An end-to-end case study of personalized warfarin dosing*, 23rd {USENIX} Security Symposium ({USENIX} Security 14), 2014, pp. 17–32.

[23] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov, *Property inference attacks on fully connected neural networks using permutation invariant representations*, Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (New York, NY, USA), CCS '18, Association for Computing Machinery, 2018, p. 619–633.

[24] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing, *CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy*, International Conference on Machine Learning, 2016, pp. 201–210.

[25] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio, *Deep learning*, vol. 1, MIT press Cambridge, 2016.

[26] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Advances in neural information processing systems, 2014, pp. 2672–2680.

[27] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, *Explaining and harnessing adversarial examples*, 2015.

[28] Inken Hagestedt, Yang Zhang, Mathias Humbert, Pascal Berrang, Haixu Tang, XiaoFeng Wang, and Michael Backes, *MBeacon: Privacy-preserving beacons for dna methylation data.*, NDSS, 2019.

[29] James A Hanley and Barbara J McNeil, *The meaning and use of the area under a receiver operating characteristic (ROC) curve.*, Radiology **143** (1982), no. 1, 29–36.

[30] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro, *LOGAN: Membership inference attacks against generative models*, Proceedings on Privacy Enhancing Technologies **2019** (2019), no. 1, 133–152.

[31] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright, *Privacy-preserving machine learning as a service*, Proceedings on Privacy Enhancing Technologies **2018** (2018), no. 3, 123–142.

[32] Benjamin Hilprecht, Martin Härterich, and Daniel Bernau, *Reconstruction and membership inference attacks against generative models*, 2019.

[33] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig, *Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays*, PLoS Genet **4** (2008), no. 8, e1000167.

[34] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel, *Chiron: Privacy-preserving machine learning as a service*, 2018.

[35] Paul Irolla and Gregory Châtel, *Demystifying the membership inference attack*, 2019 12th CMI Conference on Cybersecurity and Privacy (CMI), 2019, pp. 1–7.

[36] Roger Iyengar, Joseph P. Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang, *Towards practical differentially private convex optimization*, 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 299–316.

[37] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong, *MemGuard: Defending against black-box membership inference attacks via adversarial examples*, Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (New York, NY, USA), CCS '19, Association for Computing Machinery, 2019, p. 259–274.

[38] Kam-Chuen Jim, C Lee Giles, and Bill G Horne, *An analysis of noise in recurrent neural networks: convergence and generalization*, IEEE Transactions on neural networks **7** (1996), no. 6, 1424–1438.

[39] Daniel Jurafsky and James H Martin, *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*.

[40] Daniel Kifer, Adam D. Smith, and Abhradeep Thakurta, *Private convex optimization for empirical risk minimization with applications to high-dimensional regression*, COLT, 2012.

[41] Mateusz Kozinski, Loïc Simon, and Frédéric Jurie, *An adversarial regularisation for semi-supervised training of structured output neural networks*, 2017.

[42] Yann LeCun and Corinna Cortes, *MNIST handwritten digit database*, 2010.

[43] Bo Li and Yevgeniy Vorobeychik, *Scalable optimization of randomized operational decisions in adversarial classification settings*, Artificial Intelligence and Statistics, 2015, pp. 599–607.

[44] Gaoyang Liu, Chen Wang, Kai Peng, Haojun Huang, Yutong Li, and Wenqing Cheng, *SocInf: Membership inference attacks on social media health data with machine learning*, IEEE Transactions on Computational Social Systems **6** (2019), no. 5, 907–921.

[45] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan, *Oblivious neural network predictions via minionn transformations*, Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 619–631.

[46] Yunhui Long, Vincent Bindschaedler, and Carl A. Gunter, *Towards measuring membership privacy*, 2017.

[47] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen, *Understanding membership inferences on well-generalized learning models*, 2018.

[48] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas, *Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave*, Proceedings of the Hardware and Architectural Support for Security and Privacy 2016 (New York, NY, USA), HASP 2016, Association for Computing Machinery, 2016, pp. 1–9.

[49] Frank McSherry and Kunal Talwar, *Mechanism design via differential privacy*, 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), IEEE, 2007, pp. 94–103.

[50] Anukrati Mehta, *A comprehensive guide to types of neural networks*, 2019.

[51] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov, *Exploiting unintended feature leakage in collaborative learning*, 2018.

[52] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov, *Exploiting unintended feature leakage in collaborative learning*, 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 691–706.

[53] Charles E Metz, *Basic principles of ROC analysis*, Seminars in nuclear medicine, vol. 8, WB Saunders, 1978, pp. 283–298.

[54] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, and Shin Ishii, *Virtual Adversarial Training: A regularization method for supervised and semi-supervised learning*, 2018.

[55] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii, *Distributional smoothing with virtual adversarial training*, 2016.

[56] Noman Mohammed, Dima Alhadidi, Benjamin CM Fung, and Mourad Debbabi, *Secure two-party differentially private data release for vertically partitioned data*, IEEE transactions on dependable and secure computing **11** (2013), no. 1, 59–71.

[57] Payman Mohassel and Yupeng Zhang, *SecureML: A system for scalable privacy-preserving machine learning*, 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 19–38.

[58] Kevin P Murphy, *Machine Learning: a probabilistic perspective*, MIT press, 2012.

[59] Vinod Nair and Geoffrey E. Hinton, *Rectified linear units improve restricted boltzmann machines*, Proceedings of the 27th International Conference on International Conference on Machine Learning (Madison, WI, USA), ICML'10, Omnipress, 2010, p. 807–814.

[60] Arvind Narayanan and Vitaly Shmatikov, *Robust de-anonymization of large sparse datasets*, 2008 IEEE Symposium on Security and Privacy (sp 2008), IEEE, 2008, pp. 111–125.

[61] Milad Nasr, Reza Shokri, and Amir Houmansadr, *Machine learning with membership privacy using adversarial regularization*, CCS '18, 2018.

[62] Milad Nasr, Reza Shokri, and Amir Houmansadr, *Comprehensive Privacy Analysis of Deep Learning: Passive and active white-box inference attacks against centralized and federated learning*, 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 739–753.

[63] Augustus Odena, *Semi-supervised learning with generative adversarial networks*, 2016.

[64] Seong Joon Oh, Max Augustin, Bernt Schiele, and Mario Fritz, *Towards reverse-engineering black-box neural networks*, 2017.

[65] David L Olson and Dursun Delen, *Advanced data mining techniques*, Springer Science & Business Media, 2008.

[66] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami, *Practical black-box attacks against machine learning*, Proceedings of the 2017 ACM on Asia conference on computer and communications security, 2017, pp. 506–519.

[67] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami, *The limitations of deep learning in adversarial settings*, 2016 IEEE European symposium on security and privacy (EuroS&P), IEEE, 2016, pp. 372–387.

[68] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron

Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay, *Scikit-learn: Machine learning in python*, Journal of Machine Learning Research **12** (2011), no. 85, 2825–2830.

[69] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro, *Knock knock, who's there? membership inference on aggregate location data*, 2017.

[70] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro, *Under the hood of membership inference attacks on aggregate location time-series*, 2019.

[71] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang, *Membership inference attack against differentially private deep learning model.*, Trans. Data Priv. **11** (2018), no. 1, 61–79.

[72] Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar, *DeepSecure: Scalable provably-secure deep learning*, Proceedings of the 55th Annual Design Automation Conference, 2018, pp. 1–6.

[73] Md Nazmus Sadat, Xiaoqian Jiang, Md Momin Al Aziz, Shuang Wang, and Noman Mohammed, *Secure and efficient regression analysis using a hybrid cryptographic framework: Development and evaluation*, JMIR medical informatics **6** (2018), no. 1, e14.

[74] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes, *ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models*, 2018.

[75] Tim Salimans and Durk P Kingma, *Weight normalization: A simple reparameterization to accelerate training of deep neural networks*, Advances in neural information processing systems, 2016, pp. 901–909.

[76] Matthias Schunter, *Intel Software Guard Extensions: Introduction and open research challenges*, Proceedings of the 2016 ACM Workshop on Software PROtection (New York, NY, USA), SPRO '16, Association for Computing Machinery, 2016, p. 1.

[77] Reza Shokri and Vitaly Shmatikov, *Privacy-preserving deep learning*, Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1310–1321.

[78] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov, *Membership inference attacks against machine learning models*, 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.

[79] Md Shamimur Rahman Shuvo and Dima Alhadidi, *Membership Inference Attacks: Analysis and mitigation*, 19th IEEE International Conference On Trust, Security And Privacy In Computing And Communications, TrustCom 2020, Guangzhou, China, December 29, 2020- January 1, 2021, IEEE, 2020, pp. 1410–1419.

[80] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate, *Stochastic gradient descent with differentially private updates*, 2013 IEEE Global Conference on Signal and Information Processing, Dec 2013, pp. 245–248.

[81] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15** (2014), 1929–1958.

[82] Stephen V Stehman, *Selecting and interpreting measures of thematic classification accuracy*, Remote sensing of Environment **62** (1997), no. 1, 77–89.

[83] Latanya Sweeney, *Weaving technology and policy together to maintain confidentiality*, The Journal of Law, Medicine & Ethics **25** (1997), no. 2-3, 98–110, PMID: 11066504.

[84] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel, *Ensemble Adversarial Training: Attacks and defenses*, 2020.

[85] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart, *Stealing machine learning models via prediction apis*, 25th {USENIX} Security Symposium ({USENIX} Security 16), 2016, pp. 601–618.

[86] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Wenqi Wei, and Lei Yu, *Effects of differential privacy and data skewness on membership inference vulnerability*, 2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA) (2019), 82–91.

[87] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei, *Demystifying membership inference attacks in machine learning as a service*, IEEE Transactions on Services Computing (2019), 1–1.

[88] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis, *The max-min hill-climbing bayesian network structure learning algorithm*, Mach. Learn. **65** (2006), no. 1, 31–78.

[89] Yevgeniy Vorobeychik and Bo Li, *Optimal randomized classification in adversarial settings.*, AAMAS, 2014, pp. 485–492.

[90] Dennis D. Wackerly, William Mendenhall III, and Richard L. Scheaffer, *Mathematical statistics with applications*, sixth edition ed., Duxbury Advanced Series, 2002.

[91] Binghui Wang and Neil Zhenqiang Gong, *Stealing hyperparameters in machine learning*, 2018 IEEE Symposium on Security and Privacy (SP) (2018), 36–52.

[92] Di Wang, Minwei Ye, and Jinhui Xu, *Differentially Private Empirical Risk Minimization Revisited: Faster and more general*, 2018.

[93] Qian Wang, Minxin Du, Xiuying Chen, Yanjiao Chen, Pan Zhou, Xiaofeng Chen, and Xinyi Huang, *Privacy-preserving Collaborative Model Learning: The case of word vector training*, IEEE Transactions on Knowledge and Data Engineering **30** (2018), no. 12, 2381–2393.

[94] Cort J Willmott and Kenji Matsuura, *Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance*, Climate research **30** (2005), no. 1, 79–82.

[95] Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, James Honaker, Kobbi Nissim, David R O'Brien, Thomas Steinke, and Salil Vadhan, *Differential Privacy: A primer for a non-technical audience*, Vand. J. Ent. & Tech. L. **21** (2018), 209.

[96] Weilin Xu, David Evans, and Yanjun Qi, *Feature Squeezing: Detecting adversarial examples in deep neural networks*, 2018.

[97] Andrew Chi-Chih Yao, *How to generate and exchange secrets*, 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), IEEE, 1986, pp. 162–167.

[98] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha, *Privacy Risk in Machine Learning: Analyzing the connection to overfitting*, 2018.

[99] Zhonghui You, Jinmian Ye, Kunming Li, Zenglin Xu, and Ping Wang, *Adversarial Noise Layer: Regularize neural network by adding noise*, 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 909–913.

[100] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex, *Differentially private model publishing for deep learning*, 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 332–349.

[101] Matthew D Zeiler and Rob Fergus, *Visualizing and understanding convolutional networks*, European conference on computer vision, Springer, 2014, pp. 818–833.

[102] Bo Zhang, Ruotong Yu, Haipei Sun, Yanying Li, Jun Xu, and Hui Wang, *Privacy for All: Demystify vulnerability disparity of differential privacy against membership inference attack*, 2020.

[103] Xinhua Zhang, *Gaussian Distribution*, pp. 425–428, Springer US, Boston, MA, 2010.

[104] 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*, Information Sciences **476** (2019), 357–372.

Vita

Candidate's full name: Md Shamimur Rahman Shuvo
Bachelor of Science in Computer Science and Engineering, Rajshahi University of Engineering and Technology, Rajshahi, Bangladesh, 2012

Publication:

1. Md Shamimur Rahman Shuvo and Dima Alhadidi, *Membership Inference Attacks: Analysis and Mitigation*, 19th IEEE International Conference On Trust, Security And Privacy In Computing And Communications, TrustCom 2020, Guangzhou, China, December 29, 2020- January 1, 2021, IEEE, 2020, pp. 1410 - 1419