2023

Comparative Analysis of Membership Inference Attacks in Federated Learning

Saroj Dayal
University of Windsor

Comparative Analysis of Membership Inference Attacks in Federated Learning

By

Saroj Dayal

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2023

Comparative Analysis of Membership Inference Attacks in Federated Learning

by

Saroj Dayal

APPROVED BY:

J. Pathak
Odette School of Business

S. Khan
School of Computer Science

D. Alhadidi, Advisor
School of Computer Science

April 24, 2023

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

I Co-Authorship

I hereby declare that this thesis incorporates material that is the outcome of my research under the supervision of Dr. Dima Alhadidi. In general, the key ideas, methodology development, software programming, validation, verification, data curation, writing, and visualization were performed by Saroj Dayal, and the co-authors' contribution was primarily through the provision of study materials. Dr. Dima Alhadidi provided conceptualization ideas, conducting research, writing - review and editing, supervision, and project management Dr. Noman Mohammed and Ali Abbasi Tadi assisted in validating the idea, review, and editing process. I am aware of the University of Windsor Senate Policy on Authorship, and I certify that I have properly acknowledged the contribution of other researchers to my thesis and have obtained written permission from each co-author(s) to include the above material(s) in my thesis. I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II Previous Publication

This thesis includes partly the original paper that has been accepted for publication: 27th International Database Engineering Applications Symposium (IDEAS2023)

Publication Title / Full Citation	**Status**
S. Dayal, D. Alhadidi, A. Tadi and N. Mohammed, Comparative Analysis of Membership Inference Attacks in Federated Learning 2023	Accepted

I certify that I have obtained written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained written permission from the copyright owner(s) to include such material(s) in my thesis. I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Given a federated learning model and a record, a membership inference attack can determine whether this record is part of the model's training dataset. Federated learning is a machine learning technique that enables different parties to train a model without the need to centralize or share their local data. Membership inference attack risks the private datasets if those datasets are used to train the federated learning model and access to the generated model is available. There is a need to study the membership inference attack in the federated learning setting. In this thesis, we empirically investigated and compared various membership inference attack approaches in a federated learning environment. We evaluated these attacks on three datasets(MNIST, FMINST, CIFAR-10) using different optimizers(SGD, RMSProp, AdaGrad) and analyzed them with and without countermeasures. The experimental results show that the membership inference approach using the prediction sensitivity approach is the worst for attackers. Additionally, among all the countermeasures, knowledge distillation has significant advantages in handling the trade-off between privacy and utility.

DEDICATION

This thesis is dedicated to everyone who believes it is never too late to do better.

ACKNOWLEDGEMENTS

I would like to sincerely express my most profound gratitude towards my supervisor Dr.Dima Alhadidi, whose input helped me immensely. With her input, I was able to look at my research with a different perspective and a more critical eye.

Secondly, I would like to thank Dr. Noman Mohammed and Ali Abbasi Tadi for assisting in validating the idea, reviewing, and editing process.

Then, I would like to express my gratitude to my thesis committee members, Dr. Shafaq Khan and Dr. Jagdish Pathak, for their beneficial advice and suggestions for my thesis.

I would also like to thank my family members, Sohani Dayal, Prabhu Dayal, Anand Deshwal, Abhishek Dayal, Suman Deswal, and Rajbir Singh Deswal, for encouraging me and constantly supporting me throughout my master's degree.

I humbly extend my thanks to the School of Computer Science and all concerned people who helped me in this regard.

TABLE OF CONTENTS

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION **III**

ABSTRACT **V**

DEDICATION **VI**

ACKNOWLEDGEMENTS **VII**

LIST OF TABLES **X**

LIST OF FIGURES **XI**

LIST OF ABBREVIATIONS **XIII**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Objective . . . 2
1.3 Contributions . . . 4
1.4 Thesis Organization . . . 5

2 Related Work **6**
2.1 MIA in Centralized Learning . . . 6
2.2 MIA in Federated Learning . . . 7
2.3 Defense against MIA in Centralized Learning . . . 8
2.4 Defense against MIA in Federated Learning . . . 9
2.5 Conclusion . . . 10

3 Membership Inference Attack techniques **12**
3.1 Shokri et al.'s MIA . . . 12
3.1.1 Target Model . . . 13
3.1.2 Shadow Model . . . 14
3.1.3 Attack Model . . . 15
3.2 Salem et al.'s MIA . . . 17
3.3 Prediction Sensitivity MIA . . . 17
3.3.1 Jacobian Matrix Approximation . . . 19
3.3.2 Membership Inference . . . 20
3.4 Conclusion . . . 21

4 Countermeasures **22**
4.1 Dropout . . . 22
4.2 Gaussian Dropout . . . 23
4.3 Monte Carlo Dropout . . . 24

4.4 Gaussian Noise . . . 25
4.5 Batch Normalization . . . 26
4.6 Masking . . . 27
4.7 Activity Regularization . . . 28
4.8 Differential Privacy . . . 28
4.9 Knowledge Distillation . . . 29
4.10 Conclusion . . . 30

5 Performance Analysis **31**
5.1 Experimental Setup . . . 31
5.1.1 Datasets . . . 31
5.1.2 Dataset Preprocessing . . . 34
5.1.3 Model Architecture . . . 35
5.1.4 Training Setting . . . 36
5.1.5 Evaluation Metrics . . . 36
5.1.6 Comparison Methods . . . 36
5.2 Experimental Results . . . 38
5.2.1 CL vs FL . . . 38
5.2.2 Analysis of FL Model Accuracy . . . 40
5.2.2.1 FL model accuracy without countermeasure . . . 41
5.2.2.2 FL model accuracy with countermeasures . . . 45
5.2.3 Analysis of the Attack Recall . . . 46
5.2.3.1 Attacks without countermeasure . . . 46
5.2.3.2 Attacks with countermeasure . . . 47
5.2.3.3 Privacy and Utility . . . 48
5.3 Conclusion . . . 48

6 Conclusion **53**

REFERENCES **55**

VITA AUCTORIS **61**

LIST OF TABLES

2.4.1 Summary of the related work 11

5.1.1 General information of the datasets 34
5.1.2 Size of the dataset to train and test in the FL environment 35
5.1.3 Number of participants and size of the dataset to train and test participants in the FL environment 35
5.1.4 General information of the attacks 38
5.2.1 CL model accuracy 40
5.2.2 CL Attack Recall 41
5.2.3 FL model accuracy 45
5.2.4 FL Attack Recall 47

LIST OF FIGURES

1.1.1 Centralized learning environment 3

3.1.1 Overview of Shokri et al.'s MIA [46] 13
3.1.2 Overview of Shokri et al. [46]'s inference technique 16
3.3.1 Overview of Prediction Sensitivity inference technique [34] 19

4.1.1 Network without and with dropout [47] 23
4.2.1 Network with Gaussian dropout 24
4.3.1 Monte Carlo dropout 25
4.4.1 MNIST images by various amounts of Gaussian noise [14]. (a) Without noise. (b) Gaussian noise with 0.1 STD. (c) Gaussian noise with 0.5 STD. (d) Gaussian noise with 1.0 STD. 26
4.5.1 Batch normalization layers [21] 27
4.6.1 Masking on MNIST images [50] 28
4.9.1 A generic illustration of knowledge distillation [13] 30

5.1.1 Visualization of MNIST Dataset [22] 32
5.1.2 Visualization of FMNIST Dataset [52] 33
5.1.3 Visualization of CIFAR-10 Dataset [31] 34
5.1.4 Visualization of model architecture 37
5.1.5 Overview of a FL system 38
5.2.1 CL vs FL model accuracy on SGD optimizer - MNIST 39
5.2.2 CL vs FL Attack 1 recall on SGD optimizer - MNIST 40
5.2.3 CL vs FL Attack 2 recall on SGD optimizer - MNIST 42
5.2.4 CL vs FL Attack 3 recall on SGD optimizer - MNIST 42
5.2.5 CL vs FL Attack 4 recall on SGD optimizer - MNIST 43
5.2.6 FL model accuracy on all optimizer - MNIST 43
5.2.7 FL model accuracy on all optimizer - FMNIST 44
5.2.8 FL model accuracy on all optimizer - CIFAR-10 44

5.2.9 A comparison of the four attacks using SGD optimizers with and without countermeasures - MNIST 46
5.2.10 A comparison of the four attacks using Adagrad optimizers with and without countermeasures - MNIST 48
5.2.11 A comparison of the four attacks using RMSProp optimizers with and without countermeasures - MNIST. 49
5.2.12 A comparison of the four attacks using SGD optimizers with and without countermeasures - FMNIST 50
5.2.13 A comparison of the four attacks using Adagrad optimizers with and without countermeasures - FMNIST 50
5.2.14 A comparison of the four attacks using RMSProp optimizers with and without countermeasures - FMNIST 51
5.2.15 A comparison of the four attacks using SGD optimizers with and without countermeasures - CIFAR-10 51
5.2.16 A comparison of the four attacks using Adagrad optimizers with and without countermeasures - CIFAR-10 52
5.2.17 A comparison of the four attacks using RMSProp optimizers with and without countermeasures - CIFAR-10 52

LIST OF ABBREVIATIONS

ML	Machine Learning
CL	Centralized Learning
FL	Federated Learning
MIA	Membership Inference Attack
MCD	Monte Carlo Dropout
BN	Batch Normalization
GD	Gaussian Dropout
GN	Gaussian Noise
AR	Activity Regularization
DP	Differential Privacy
KD	Knowledge Distillation
SGD	Stochastic Gradient Descent
RMSProp	Root Mean Squared Propagation
Adagrad	Adaptive Gradient

CHAPTER 1

Introduction

In this chapter, we briefly address the motivation, objectives, and contributions of this thesis work.

1.1 Motivation

Machine Learning (ML) is a field of study that tries to make computers learn and behave as humans do by providing data and information in the form of observations and interactions with people in the actual world [23]. ML is getting more attention from researchers in many fields because of the accessibility of a vast amount of data and technological advancements [39]. Leveraging ML and enormous amounts of data can provide a potential solution to the critical issues faced in many fields of study. ML can provide improved data pattern differentiation by efficiently revealing more data.

Managing vast amounts of data has been challenging to maintain the efficiency and scalability of the ML algorithms. As a result, it is vital to investigate learning techniques that can handle distributed datasets. Traditional centralized ML techniques are not the best choice in these situations since they demand the transfer and processing of data at a centralized server, which may not be practical given the accessibility of private data. The participants are connected to the centralized server to upload their data, as shown in Figure 1.1.1. The centralized server manages all computational activities necessary to train the data when the participants upload their local data to it [15]. Centralized training is resource-efficient for the participants since

they are relieved of the computation-related duties that demand more resources [15]. However, because a centralized server could be malicious or compromised by attackers, participants' information confidentiality is seriously threatened. A large amount of data uploading might also increase participants' time to communicate with the centralized server [15]. To achieve the best model possible, more data is needed. Now, what if those participants do not want to share their data but still want to aggregate it to build an excellent ML model? Decentralized learning approaches are needed, where all the private data is held locally, and only locally-trained models are sent to the central server. There comes Federated Learning.

Federated Learning (FL) has expanded into a distributed ML paradigm since Google first proposed it in 2017 [35]. Its objective is to improve privacy by allowing data owners to successfully train a model on their shared training data with the support of a central server without revealing their potentially sensitive data to either the central server or each other. While FL uses the enormous amount of data now accessible at edge devices, it increases data privacy by allowing data to be kept locally at the clients. This is very important when dealing with sensitive and personal data, such as in the health industry, where ML is gaining more interest, especially with the existence of legal obligations like the GDPR [2] and HIPAA [1].

A Centralized Learning (CL) model raises privacy concerns because participants must upload their data to the central server to train the model. However, FL enables participants to train models on local data collaboratively without revealing sensitive information to the central server. As a result, the FL model could be wisely applied to any domain because it protects user-sensitive data that does not need to be sent to the server. Rather than traditional ML, FL is the best solution when massive amounts of data and participants' private information are involved.

1.2 Objective

Despite the advantages, FL is vulnerable to several inference attacks; one such attack is a membership inference attack (MIA), which attempts to find information about

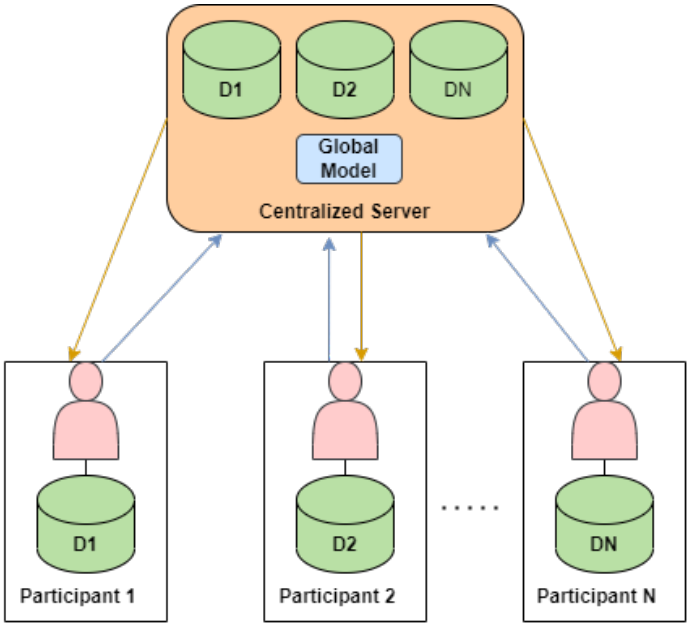


Fig. 1.1.1: Centralized learning environment

the training data used. In this inference attack, the attacker can infer whether the data is part of the training dataset. This is actually against FL's primary design purpose, which is protecting data privacy.

A successful MIA can pose a severe risk to private datasets, like healthcare data, if these are used to train an FL model and access to the resulting model is made available to the general public. For instance, if a FL model is trained using data gathered from people with a particular disease, an attacker may learn the patient's health status by being aware that the patient's data are included in the model training data. As a result, the individual's privacy may be compromised. This is also essential for ML services like Google Prediction because they would lose many clients if they could not ensure the privacy of training data. MIA has been successfully executed on healthcare data [16]. The success of privacy attacks on ML models depends on the training data nature and the type of modeling technique. A FL system with several participants and datasets can be complex, and various factors can significantly influence the effectiveness of privacy attacks. Therefore, it is equally crucial to study trends in inference risk with parameter variations since some configurations can be more vulnerable than others. Investigating those configurations and evaluating the

associated risks enable one to make informed decisions regarding the system design choices.

Our main objective is to investigate the membership inference attack in the FL setting where the attacker tries to know if the data is part of the training dataset and investigate the effectiveness of different countermeasures against this attack.

1.3 Contributions

Related work investigates the MIA in the centralized setting where one data owner owns data. There is a desideratum to study the MIA in the FL setting. In this thesis, we analyze different techniques [46, 44, 34] of the MIA, initially proposed in the centralized setting, in the FL setting together with the effectiveness of the countermeasures to mitigate those attacks. To the best of our knowledge, this thesis is the first to conduct such an experiment. The contributions of our thesis work are summarized below.

- We implemented four attacks on the FL setting [46, 44, 34].
- We compared the FL model accuracy and attack recall with three optimizers: Stochastic Gradient Descent (SGD) [18], Root Mean Squared Propagation (RMSProp) [49], and Adaptive Gradient (Adagrad) [36], on three real-world datasets, MNIST [22], Fashion-MNIST (FMNIST) [52], and CIFAR-10 [31].
- We investigated the impact of various countermeasures on FL model accuracy and attack recall in the FL setting. We experimentally showed that Knowledge Distillation (KD) and Monte-Carlo Dropout (MCD) are the best mitigations in the FL environment. We also showed that Shokri et al.'s attack [46] is the best choice for attackers, whereas Liu et al.'s attack [34] is the worst for them.
- We conducted a comparative study in terms of FL model accuracy and attack recall, both of which are important for the design of future models.

1.4 Thesis Organization

The rest of the thesis paper is structured as follows. In Chapter 2, we introduced the related work. We explained various membership inference techniques in Chapter 3. Countermeasures are explained in Chapter 4. The experimental setup and results are provided and analyzed in Chapter 5. Finally, we conclude our work in Chapter 6.

CHAPTER 2

Related Work

In this chapter, we outlined the related work of the Membership Inference Attack (MIA) in Centralized Learning (CL) and Federated Learning (FL). The related work targeting MIA in CL and FL is summarized in Table 2.4.1.

2.1 MIA in Centralized Learning

Shokri et al. [46] performed the first MIA against ML models to determine the presence of a data sample while training a model from a black-box target model. They [46] made two necessary assumptions to conduct the MIA against ML models. The first is to create multiple shadow models with the same structure as the target model to imitate the target model's behavior. The second assumption relates to the data generated for the shadow model training. They used the same distribution as the target model's training data to generate the data to train the shadow models. They [46] created several attack models, each representing a specific prediction class of the target model. When used for training, the data are labeled "in" and labeled "out" when used for testing. The label "in/out" and the probability score vector are applied to train the attack model. The attack model is a binary classifier that infers the target model's members based on the probability score vector of the target model output since the attack model's final class label is either "in" or "out".

Afterward, Salem et al. [44] expanded the scope of the MIA by relaxing the assumptions of Shokri et al. [46]. First, they relaxed the assumption regarding the number of shadow models by using a single model to produce the data for the attack

model training. However, having a single shadow model decreases the data to train the attack model. Attack accuracy is affected and is lower than when using multiple shadow models. Later, they showed that the structure and distribution of the data used to train the shadow model must differ from those of the target data. They also revealed that only one shadow and attack model would be enough to perform MIAs. However, this work requires previous knowledge of the target model's training data. Finally, they presented the attack without any shadow model training by relying just on the output of the target model queried with the target data points.

Lan Liu. [34] looked into feature space perturbation and presented Aster, a brand-new MIA for ML models, solely using a black-box API. They find that the training data's sensitivity to a fully trained ML model is lower than that of the untrained data. They [34] utilize the Jacobian matrix to quantify the link between the target model's prediction and the target sample's feature value to get the sensitivity and perform MIAs by distinguishing the sensitivity values of different data samples. Experiments on various target models and datasets confirm that Aster outperforms other MIAs without the previous details about the target model and the statistical information of its training dataset.

2.2 MIA in Federated Learning

Nasr et al. [38] presented the attack in both CL and FL environments. They derived a membership score for the target model based on the model activations, predictions, and losses in the CL setting. This work requires a prior understanding of the target model's structure and parameters. They attacked white-box deep learning models to infer the data records. The attacker used the privacy exposure of the stochastic gradient descent of the deep learning model to collect the data sample's features and conduct the MIA.

Researchers have discovered that attackers can still retrieve a participant's private information in the FL setting by listening to the messages it exchanges with the server. Passive MIA [38] allows the attacker to determine whether the participant

owns a specific data instance. In the FL setting, Nasr et al. [38] demonstrate that adversarial participants can run active MIA against other participants, even when the global model achieves high prediction accuracy.

2.3 Defense against MIA in Centralized Learning

Related work carried out many defense mechanisms to reduce MIA. Conti et al. [20] proposed a label-only MIA against Graph Neural Networks (GNNs) inspired by Convolutional Neural Networks (CNNs). They examine how the attack performance of label-only MIA is affected by the sampling technique, the model selection strategy, and the degree of overfitting. Then, they consider scenarios where assumptions regarding the attacker's additional dataset (shadow dataset) and other information about the target model are relaxed. Finally, they examined the effectiveness of potential defenses, such as dropout, regularization, normalization, and jumping knowledge.

To better handle the trade-off between privacy and utility, Zheng et al. [56] propose Complementary Knowledge Distillation (CKD) and Pseudo-Complementary Knowledge Distillation (PCKD) as two deep learning algorithms. In CKD, the transfer data of knowledge distillation all come from the private training set, but their soft targets are generated from the teacher model trained using their complementary set. With a similar idea, they propose PCKD, which reduces the training set of each teacher model and uses model averaging to generate soft targets of transfer data. PCKD uses pre-training to increase the utility of teacher models because a smaller training set results in less utility. According to experiment results, CKD and PCKD have a considerable advantage over other defense strategies, such as Distillation for membership privacy (DMP), adversarial regularisation, dropout, and Differential Privacy Stochastic Gradient Descent (DP-SGD), in addressing the privacy and utility trade-off.

Other defense methods use differential privacy mechanisms or adversarial training that struggle to balance privacy and utility. Methods based on knowledge transfer to improve model utility need unlabeled public data in the same distribution as pri-

vate data, and this requirement may not be satisfied in some scenarios. The serious privacy concerns due to the membership inference have motivated multiple defenses against MIAs, e.g., differential privacy and adversarial regularization. Unfortunately, these defenses produce ML models with unacceptably low classification performances. DMP, a novel defense against MIAs proposed by Shejwalkar et al. [45], preserves the utility of the resulting models substantially better than earlier defenses. DMP leverages knowledge distillation to train ML models with membership privacy. To train ML models with membership privacy, DMP makes use of knowledge distillation. To enhance the membership privacy of DMP, they offered a novel criterion for tuning the data used for knowledge transfer. They demonstrate that, as compared to other MIA defenses, DMP offers noticeably better tradeoffs between membership privacy and classification accuracy.

2.4 Defense against MIA in Federated Learning

FL is susceptible to a variety of MIAs. This MIA aims to uncover the data used to train the model, which has implications for participants using their data to train the shared model. FL is mainly attacked because of its unique model training mechanism. Attacks in FL aim to destroy the model's integrity, availability, and confidentiality.

Recent research on countermeasure techniques has limits in terms of ensuring privacy while reducing the loss of the model and mainly focuses on protecting the parameters. Xie et al. [54] proposed a defense method for FL against MIA. To achieve a trade-off between privacy security and the utility loss of the target model, Authors [54] suggested an adversarial method to generate the noise added to the attack features of the attack model. Before the central server distributes the confidence score vector of the global model with the participants, Authors [54] introduce noise with a specific probability in each iteration.

According to another study by Lee et al. [32], a common defense against such attacks is the differential privacy scheme, which intensifies each update with enough noise to make a recovery difficult. Unfortunately, it suffers from a considerable loss in

the FL's classification accuracy. The Digestive Neural Network (DNN) will process each participant's private data before being used to train the FL. To increase the classification accuracy of FL while reducing the accuracy of inference attacks, the DNN alters the input data, which distorts updates. This work has supported the proposed scheme's scalability by demonstrating its excellent performance on the FedAvg and FedSgd protocols.

Another paper, Su et al. [48], proposes a new privacy mechanism named the federated regularization learning model, a novel FL model that can protect data privacy from gradient leakage and black-box MIA. The proposed [48] protection scheme makes the data hard to reproduce and distinguish from predictions. A small simulated attacker network is embedded as a regularization punishment to defend against malicious attacks. They introduce a gradient modification method to secure the weight information and remedy the additional accuracy loss.

2.5 Conclusion

In this chapter, we summarize various research work in CL and FL against MIA, along with some defense approaches. Most of the MIA techniques are implemented in the CL setting, but there is a need to analyze the MIA techniques in the FL setting. We implemented these attacks [46, 44, 34], which were initially carried out in the CL environment in the FL setting. We also implemented these attacks with countermeasures in the FL setting to mitigate MIA in our experiments.

Table 2.4.1: Summary of the related work

Paper	**CL or FL**	**Attack or Defense**	**Contribution**	**Techniques**	**Limitations**
Shokri et al. [46]	CL	Attack, Defense	Determine if the record is in the model's training dataset	Target Model, Shadow Model, Attack Model	-
Salem et al. [44]	CL	Attack, Defense	Propose three adversaries and two defense.	Single Shadow model, Data not from the same distribution, No Shadow model, Dropouts, Model Stacking	Utility loss of the model
Liu et al. [34]	CL	Attack	Determine if the record is used to train the given ML model	Target Model, Jacobian Matrix, Clustering Algorithms	-
Nasr et al. [38]	CL, FL	Attack	Various categories of inference attacks against machine learning models, based on their prior knowledge	Exploiting the privacy vulnerabilities of the stochastic gradient descent algorithm	-
Conti et al. [20]	CL	Attack, Defense	Propose a label-only MIA against Graph Neural Networks (GNNs) inspired by Convolutional Neural Networks (CNNs)	Model selection strategy, Dropout, Regularization, Normalization, and Jumping knowledge	None of those four defenses prevent the attack completely
Zheng et al. [56]	CL	Defense	Propose complementary knowledge distillation (CKD) and pseudo-complementary knowledge distillation (PCKD)	Knowledge Distillation	Assumption of an additional dataset with a close distribution to the private dataset
Shejwalkar et al. [45]	CL	Defense	Propose distillation for membership privacy (DMP)	Knowledge Distillation	Low classification performances
Xie et al. [54]	FL	Defense (server)	Propose Fedefend	Central server adds noise to the confidence score vector of the model before sharing the global model with the participants	Utility loss of the model
Lee et al. [32]	FL	Defense (participants)	Propose Digestive Neural Network (DNN)	DNN modifies the input data, which results in distorting updates	Degree of noise has little impact on the classification accuracy
Su et al. [48]	FL	Defense (participants)	Propose regularization punishment to prevent malicious attacks	Used gradient modification algorithm in the training procedure	Model performs slightly worse on test accuracy than the baseline

CHAPTER 3

Membership Inference Attack techniques

In this chapter, we discussed the techniques used to carry out membership inference attacks as described by Shokri et al. [46], Salem et al. [44] and Liu et al. [34]. We carried out these attacks in the federated learning setting in our thesis.

3.1 Shokri et al.'s MIA

ML models behave differently with the data they see for the first time than the data they are trained on. An attacker uses 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. [46] constructed multiple shadow models to mimic the target model's behavior and used the output of the shadow models to train the attack model. The target model has a private dataset containing the labeled data records (x_i, y_i) where x_i is a feature of the data point, and y_i is an actual output 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 predicted label for the data record is the class with the highest probability. Shadow models are created to mimic the target model. However, the shadow and target models are trained using separate datasets, such that $D_{target} \cap D_{Shadowi} = \phi$. The inputs and outputs of the shadow models are used to train the attack models.

To construct the attack model's training data, the training set of the shadow

models is queried and labeled "in" and the test dataset of the shadow models is also queried and labeled "out". As a result, the attack model's training dataset contains the trained dataset's output probabilities with the label "in" or "out". The output of the target model infers the members using the attack model. The overview of the MIA is shown in Figure 3.1.1. To understand the MIA, let us discuss the following three models:

- Target model
- Shadow model
- Attack model

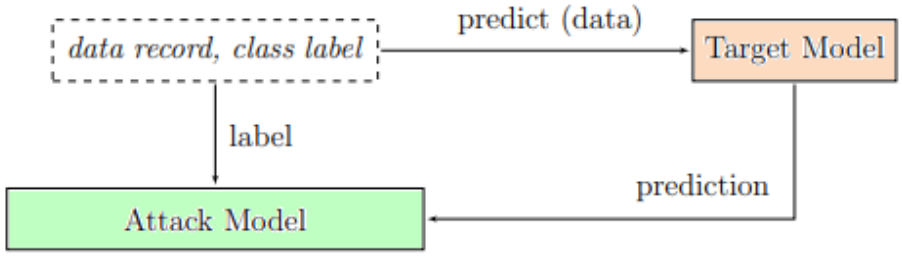


Fig. 3.1.1: Overview of Shokri et al.'s MIA [46]

3.1.1 Target Model

The target model captures the relationship between the content of the data records and their actual labels by taking the data records as input and, after training, outputs the prediction vector of probabilities. The class with the highest confidence level is chosen as the predicted label for the data record. Assume that D_{Target}^{Train} be the private training dataset of the target model (M_{Target}) where (x_i, y_i) are the labeled data records. In this labeled data record, (x_i) represents the input to the target model, whereas (y_i) represents the actual label of the data record, which takes the values from a set of classes of size C_{Target}. The target model's output (M_{Target}) is a vector of probabilities of size C_{Target} where the elements range from 0 to 1, and the sum of the vector is equal to 1. The accuracy of the target model is evaluated 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. MIA takes advantage of the notion that ML models respond differently when presented with data different from that on which they were trained. The attacker's goal is to construct an attack model that can recognize the behavior 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.

3.1.2 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 mimic the target model's behavior and create the data required for the attack model's training. Multiple shadow models are created to train the attack model similar to the target model. The attacker creates n number of shadow models $M^i_{Shadow}()$ where each shadow model i is trained on dataset D^i_{Shadow}. The attacker first splits its dataset D^i_{Shadow} into two sets D^{iTrain}_{Shadow} and D^{iTest}_{Shadow}, such that,

$$D^{iTrain}_{Shadow} \cap D^{iTest}_{Shadow} = \phi \tag{1}$$

The attacker then trains each shadow model M^i_{Shadow} using the training set D^{iTrain}_{Shadow} and tests the same using D^{iTest}_{Shadow} set of the data. D^i_{Shadow} dataset 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. [46] considered the worst case for the attacker such that the dataset used for training the shadow models is disjoint from the private dataset

used to train the target model such that,

$$\forall D_{Shadow}^{i^{Train}} \cap D_{Shadow}^{i^{Test}} = \phi \tag{2}$$

The attack model is trained to recognize differences in shadow models' behavior when these models operate on inputs from their 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. The overview of the MIA is shown in Figure 3.1.2.

3.1.3 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. [46] used multiple shadow models, which behave similarly to the target model. For each shadow model, if a given record is in the training set, it is labeled as "in" whereas if it is in the testing set, it is labeled as "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. During the prediction of the attack model, its input comprises a correctly labeled record and a prediction vector of probabilities generated by the target model for the corresponding record. Since the Attack's goal is decisional membership inference, the attack model is a binary classifier with two output classes, "in" and "out".

The training data for the attack model comes from the inputs and outputs of the shadow models. Each shadow model is queried using all the records of its training dataset and the disjoint test dataset of the same size to get the output. First, a particular shadow model is queried with its training dataset to get the output vectors

labeled “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 labeled ”out” and added to the attack model’s training set. The exact process is repeated for all the shadow models to construct the training dataset of the attack model. This dataset shows 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 ith shadow model. For all training data of the ith shadow model, the prediction vector $Y = M_{Shadow}^{i}(x)$ is generated. The record $(y,\ Y,\ in)$ is added to the training dataset of the attack model where the prediction vector is $Y = M_{Shadow}(x)$. Again, let $D_{Shadow}^{Test^i}$ be the test dataset disjoint from the training dataset of the ith shadow model. Then, $\forall$(x,y) $\in D_{Shadow}^{Test^i}$, the prediction vector $Y = M_{Shadow}^{i}(x)$ is computed, and the record $(y,\ Y,\ out)$ is added to the training dataset of the attack model. The same procedure is applied to all the shadow models to get the training set, D_{Attack}^{Train} for the attack model. Shokri et al. [46] splits the dataset D_{Attack}^{Train} into C_{Target} partitions, each representing an independent class of the target model, and trains a separate model for each class that predicts the “in” or “out” status of x.

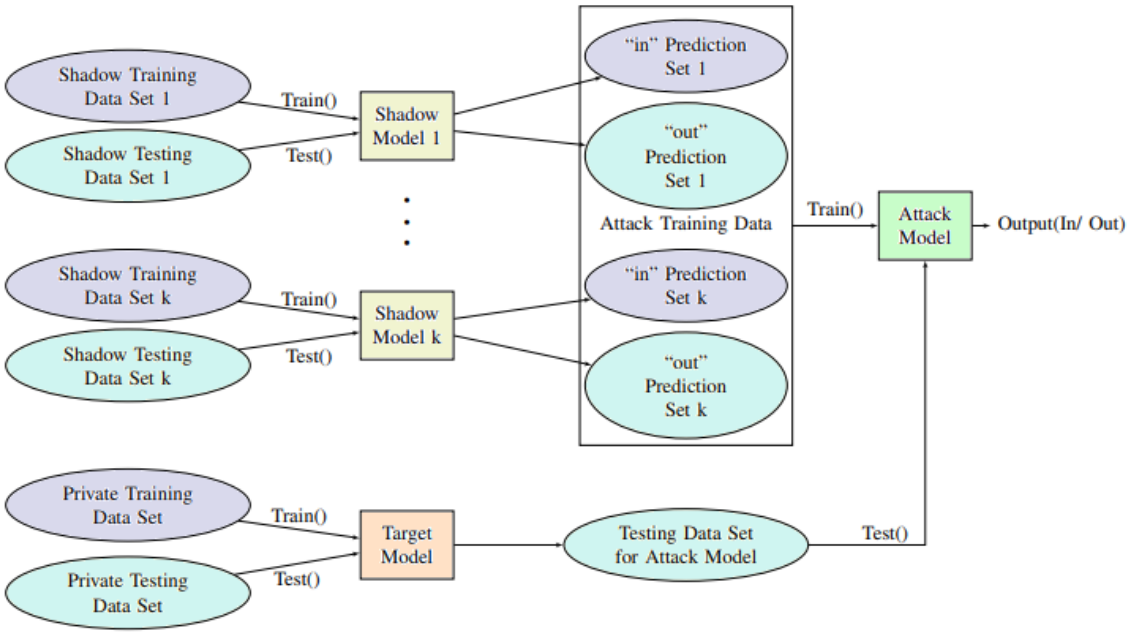


Fig. 3.1.2: Overview of Shokri et al. [46]’s inference technique

3.2 Salem et al.'s MIA

The early demonstrations made by Shokri et al. [46] of the feasibility of the MIA have many assumptions about the attacker, including the use of several shadow models, familiarity with the target model's structure, and knowledge of a dataset from the same distribution as the target model's training data. Salem et al. [44] relaxed all these fundamental assumptions, demonstrating that such attacks are widely accurate at low cost and pose a more severe risk than previously believed.

The dataset used by the first attacker is drawn from the same distribution as the training set for the target model. Here, the focus is on relaxing the shadow models' assumption. Instead of several shadow models, just one mimics the behavior of the target model. Using one shadow model dramatically lowers the cost of carrying out the MIA since shadow models are created through MLaaS, which applies the pay-per-query business model.

The data for the second attacker does not come from the same distribution as the training set for the target model. Also, the attacker does not know the structure of the target model. Compared to the prior attack scenario, this one is more realistic. Salem et al. [44] suggest a data transferring attack for MIA. Here, the shadow model is merely used to record the membership status of data points in an ML training set, not to imitate the behavior of the target model.

3.3 Prediction Sensitivity MIA

The key idea of this attack is that the training data of a fully trained ML model usually have lower prediction sensitivities than that of the non-training data, i.e., testing data. Less sensitivity indicates that when perturbing a training sample's feature value in the corresponding feature space, the prediction of the perturbed sample received from the target model tends to be consistent with the original prediction. This work [34] measures the prediction sensitivity with the Jacobian matrix, which reflects the relationship between each feature's perturbation and the related predic-

tion's change, and then clusters the samples based on their prediction sensitivity. This can breach the membership privacy of the target model's training data with no previous knowledge about the target model or its training data.

The goal is to determine whether a sample was used to train the target model. This does not assume previous information about the target model or training data. In other words, the attacker only has the black-box API of the target model.

In a more realistic situation, an attacker only has black-box access to the target model. This assumption restricts the Attack since the attacker does not obtain information about the target model's structure, type, parameters, training algorithm, settings, or training data statistics. Thus, the only information to obtain from the target model is the prediction probability vector of a given input. The only interaction allowed between the attacker and the target model M is to query M with a sample x and then get the prediction output: $M(x) = [y^1, y^2, \ldots, y^c, \ldots, y^{|C|}]$ where C is the set of class labels that the target model and y^c is the probability that the input sample belongs to class c. The only detail the attacker can receive from the target model M is the prediction probability vector. Given the target model M and a target sample x_t, the attacker tries to decide whether x_t is from M's training set: $A(x_t, M) \rightarrow In/Out$. The attacker is abstracted by A. Its inputs are M and x_t; its output is either In or Out. The output In means that x_t is from M's training set, while Out has the opposite meaning. Given a target sample x_t and the target model M with only black-box access, the focus is on whether x_t was used to train M. First, the Jacobian matrix is approximated for x_t and derives the prediction sensitivity regarding M. Then clusters the target samples according to the sensitivity, determining whether x_t is from M's training set. Therefore, there are primarily two steps to execute this MIA, as shown in Figure 3.3.1:

- Jacobian Matrix Approximation
- Membership Inference

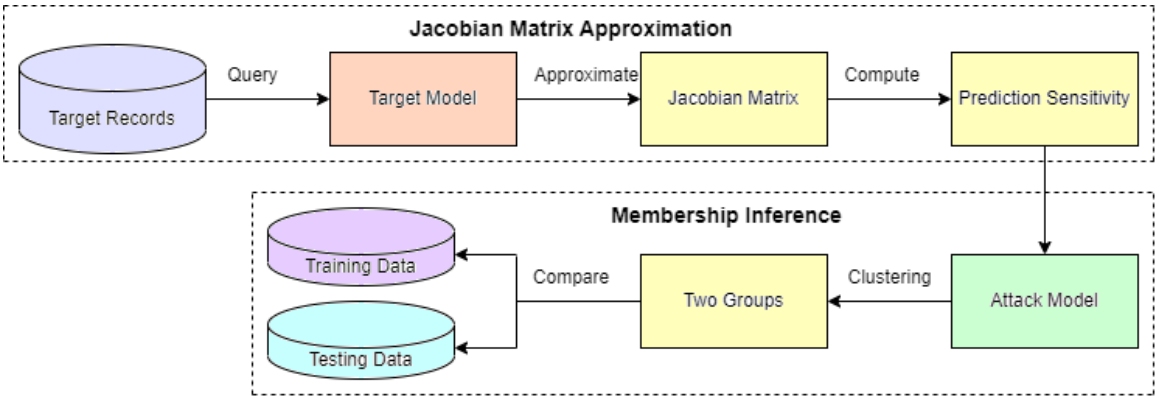


Fig. 3.3.1: Overview of Prediction Sensitivity inference technique [34]

3.3.1 Jacobian Matrix Approximation

For the target sample, input the target model M, and the target model will return the prediction probability vector. Each value in the vector defines the probability that the target sample is predicted to belong to the related class by the target model. The sum of the probability vector should be equal to 1.0. Then, standardize the procedure: $y = M(x)$, where x is the input sample, and y is the predicted probabilities vector. M here denotes the target model's black-box API.

An ML model can be considered as a function $M : \mathbb{R}^n \to \mathbb{R}^m$ that maps n-dimensional vector $x \in \mathbb{R}^n$ to m-dimensional output $y \in \mathbb{R}^m$. Then the Jacobian matrix of M is determined to be an m×n matrix, whose element located in i-th row and j-th column is $J_{ij} = \frac{\partial y_i}{\partial x_j}$ ($i \in [1, 2, \ldots, n]$ and $j \in [1, 2, \ldots, m]$):

$$J(x; M) = \left[\frac{\partial M(x)}{\partial x_1} \cdots \frac{\partial M(x)}{\partial x_n} \right] = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} \tag{3}$$

where $y = M(x)$. The input sample is $x = [x_1, x_2, \ldots, x_n]$, and the corresponding prediction is $y = [y_1, y_2, \ldots, y_m]$. $\frac{\partial y_i}{\partial x_i}$ describes the relationship between the change of the input sample's i-th feature value and the change of the prediction probability that this sample belongs to j-th class. From the Jacobian matrix's definition, the

matrix contains a series of first-order partial derivatives. Even the internal states are not present in the function M; derivatives can still be approximated by computing the numerical differentiation with the following equation:

$$\frac{\partial y_j}{\partial x_i} \approx \frac{M(x+\epsilon) - M(x-\epsilon)}{2\epsilon} \tag{4}$$

where ϵ is a small value added to the i-th feature value of the input sample. For the target sample x_t whose membership property is interested, add ϵ to (resp. minus ϵ from) the i-th feature value of the target sample and get two altered samples. Then query the target model with the two altered samples, and derive the partial derivatives of i-th feature concerning the target model: $\frac{\partial M(x)}{\partial x_i} = \left[\frac{\partial y_1}{\partial x_i}, \frac{\partial y_2}{\partial x_i} \ldots, \frac{\partial y_m}{\partial x_i}\right]$. Repeat the above process for each feature in x consecutively and combine the partial derivatives into the Jacobian matrix. Now the approximation of the Jacobian matrix is estimated, which is defined as $J(x; M)$ for clarity. Then we need to extract the prediction sensitivity of the target sample for the target model. Novak et al. [40] use the L-2 norm of $J(x; M)$ to describe the prediction sensitivity for the target sample. For a mn matrix A, the L-2 norm of A can be computed by:

$$||A||_2 = (\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2)^{\frac{1}{2}} \tag{5}$$

where i and j are the row number and the column number of the matrix element $a_{i,j}$, respectively.

3.3.2 Membership Inference

With the norm of the Jacobian matrix approximation $J(x; M)$, it is possible to decide whether the target sample is from the target model's training set. This work is based on the remark that ML models usually exhibit less sensitivity concerning the prediction behavior of the training data. The prediction sensitivity of samples from the training set is typically lower than that of the samples from the testing set. It comes inherently to leverage an unsupervised clustering method to group a set of target records into 2 clusters and then choose the cluster with lower mean sen-

sitivity than the members of the M's training set. However, the experiments show that performance is even better when the cluster number is more significant. Therefore, different numbers of clusters are tried, and setting the number 6 is based on the average attack performance against various models and datasets. Then, several clustering algorithms were compared, and the spectral clustering algorithm was used to construct the attack clustering model. At the inference stage, cluster the samples into three or more groups and then order the groups by the average norm. The groups with more minor average norms are from the target model's training set, and others are not.

3.4 Conclusion

This chapter summarizes three MIA techniques [46, 44, 34]. These techniques are initially implemented in a centralized learning setting. We implemented these techniques in the federated learning setting to provide a comparative analysis in our experiment.

CHAPTER 4

Countermeasures

This chapter covered countermeasures we applied to CL and FL models to mitigate MIA.

Attackers take advantage of the fact that ML models behave differently during the prediction with new data than with training data to differentiate members from nonmembers. This property is associated with the degree of overfitting, which measures the difference between training and test accuracy. The deeper the overfitting of the target model, the more vulnerable it is to MIA. In this chapter, we detailed a variety of possible countermeasures to reduce overfitting in the target model.

4.1 Dropout

Dropout (D) prevents overfitting by randomly dropping out units in a neural network and allows for the approximate efficient combination of many different neural network architectures [47]. The term "dropout" refers to dropping out units (hidden and visible) in a neural network. Dropping a unit out means temporarily removing it from the network and all its incoming and outgoing connections, as shown in Figure 4.1.1. The choice of which units to drop is random. In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or set at 0.5, which seems to be close to optimal for a wide range of networks and tasks [47]. However, for our experiments, we used TensorFlow Keras Dropout [9] to implement this countermeasure and set the optimal probability of retention at 0.2. This is implemented as a mitigation technique against MIA in

ML models in the CL setting by Salem et al. [44].

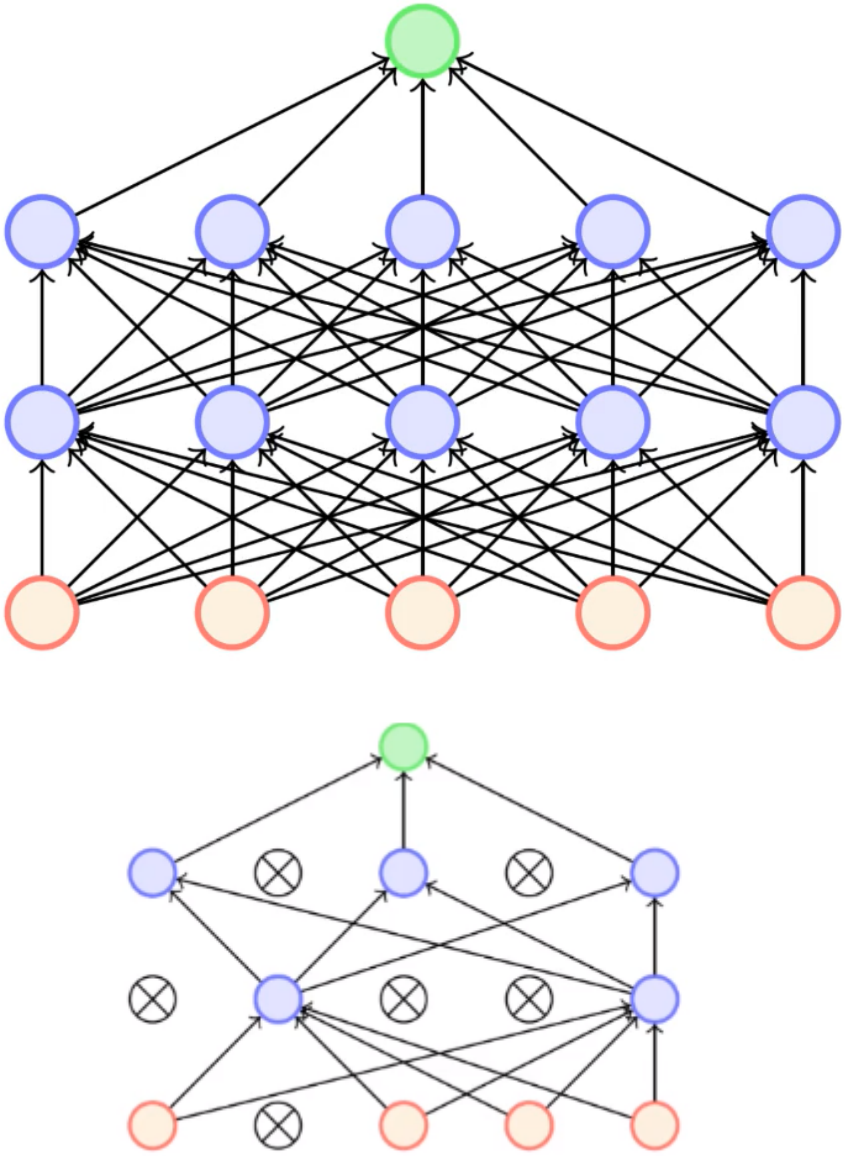


Fig. 4.1.1: Network without and with dropout [47]

4.2 Gaussian Dropout

During a forward pass, the training process aims to minimize loss. Excessive attempts to minimize this loss leads to overfitting. This random ignoring of nodes ensures that the error is calculated for the remaining nodes during training, but the weights are

updated nonetheless. However, by applying standard dropout the model is diminished. For a dropout-trained model, the time complexity for back-propagation is more, as the dropped nodes must be accounted for in the update of weights. In addition, a thinner model due to dropped nodes can risk the loss of valuable information extracted. Gaussian dropout (GD) presents a better alternative in this regard.

GD integrates Gaussian noise into a random probability of nodes. In contrast to standard dropout, nodes are not dropped entirely; instead of ignoring nodes, they are subjected to Gaussian noise, as shown in Figure 4.2.1. According to the experiments conducted by Srivatsava et al. [47], using GD resulted in less computation time because the weights did not need to be scaled each time to fit the ignored nodes, as in standard dropout. For our experiments, we used TensorFlow Keras Gaussian dropout [10] to implement this countermeasure and set the drop probability at 0.4. As a regularization layer, it is only active at training time. In the context of both CL and FL, our study is the first to use this countermeasure against MIAs.

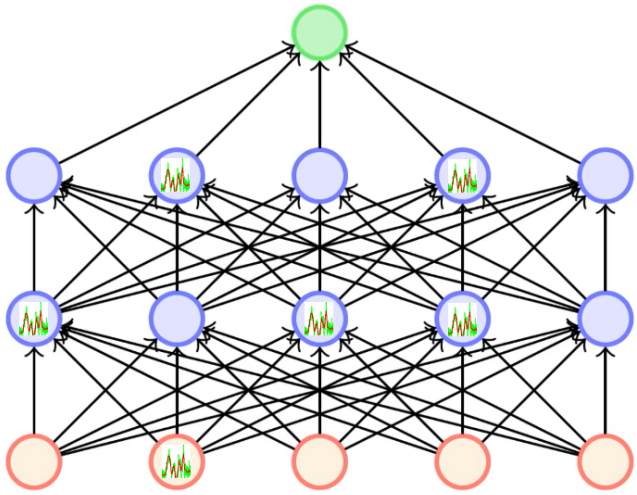


Fig. 4.2.1: Network with Gaussian dropout

4.3 Monte Carlo Dropout

Monte Carlo Dropout (MCD) is a method for capturing model uncertainty proposed by Gal and Ghahramani [28]. The different networks (with various neurons removed)

can be considered Monte Carlo samples from the space of all available models. This provides mathematical grounds for the model to reason about its uncertainty, frequently improving its performance. It works by allowing dropout during testing [41]. Then, instead of making a single prediction, it makes several, one for each model, and averages or analyses their distributions. Figure 4.3.1 shows a simple use of MCD layers. For our experiments, we used TensorFlow Keras Monte Carlo dropout [41] to implement this countermeasure and set the optimal probability of retention at 0.2. Our work is the first to implement it as a countermeasure against MIA in both CL and FL settings.

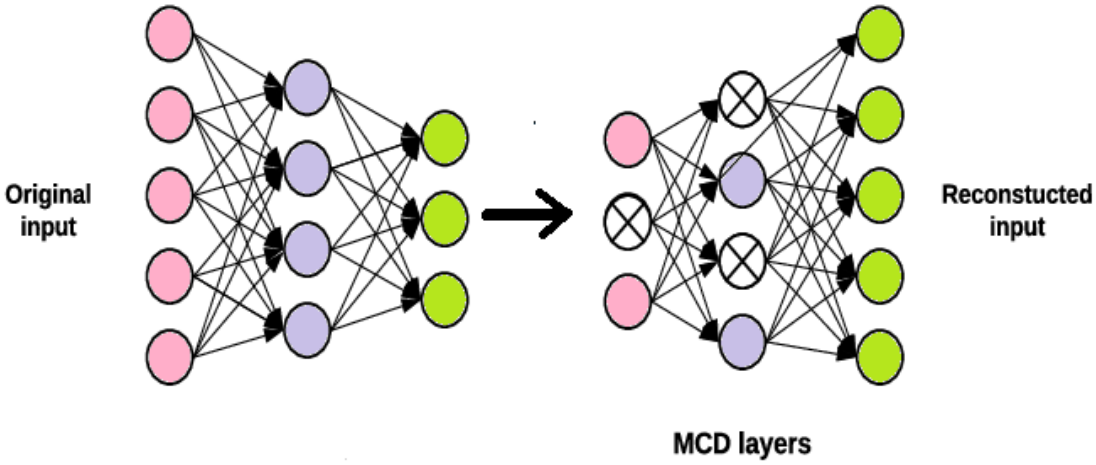


Fig. 4.3.1: Monte Carlo dropout

4.4 Gaussian Noise

Gaussian Noise (GN) is the most practical perturbation-based model for describing nonlinear effects induced by additive Gaussian noise [43]. GN is used to replace adversarial attacks to reduce the dependency of the models on specific adversarial attacks [53]. From this, we try to implement it as a countermeasure against MIA in the FL setting. Figure 4.4.1 shows examples of MNIST images by various amounts of GN. This is useful to mitigate overfitting. It is a natural choice as a corruption process for real-valued inputs. For our experiments, we used TensorFlow Keras Gaussian

noise [11] to implement this countermeasure and set the standard deviation of the noise distribution at 0.4. As a regularization layer, it is only active at training time. Our work is the first to implement it as a countermeasure against MIA in the FL setting.

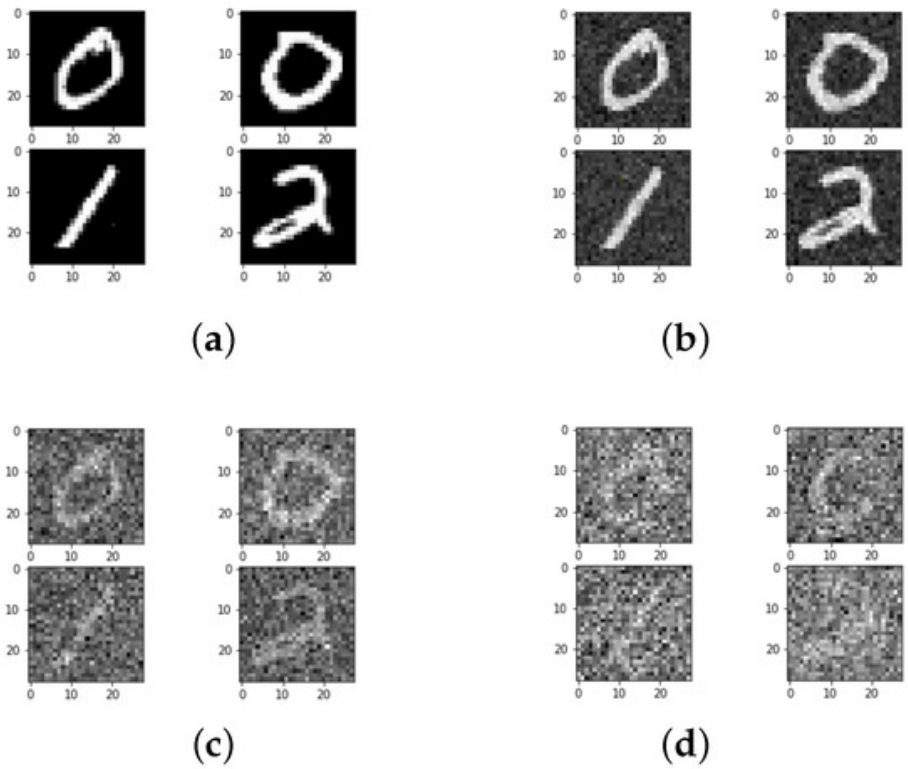


Fig. 4.4.1: MNIST images by various amounts of Gaussian noise [14]. (a) Without noise. (b) Gaussian noise with 0.1 STD. (c) Gaussian noise with 0.5 STD. (d) Gaussian noise with 1.0 STD.

4.5 Batch Normalization

Batch Normalization (BN) technique improves accuracy by normalizing activation's in intermediate layers of deep neural networks [17]. BN naturally extends this idea across the intermediate layers within a deep network [29]. However, for speed reasons, the normalization is performed across mini-batches, not the entire training set. It applies a change that holds the mean output close to 0 and the output standard deviation close to 1. It performs differently during training and inference. During

training, that is, when using fit(), the layer normalizes its output using the mean and standard deviation of the current batch of inputs. During inference, that is, when using evaluate() or predict(), the layer normalizes its output using a moving average of the mean and standard deviation of the batches it has noticed during training. As such, the layer will only normalize its inputs during inference after training on data with similar statistics to the inference data. For our experiments, we used TensorFlow Keras Batch normalization [7] to implement this countermeasure. An illustration of BN is shown in Figure 4.5.1. Normalization is used as a defense in label-only MIA, and results show that regularisation and normalization can slightly reduce average attack accuracy [20].

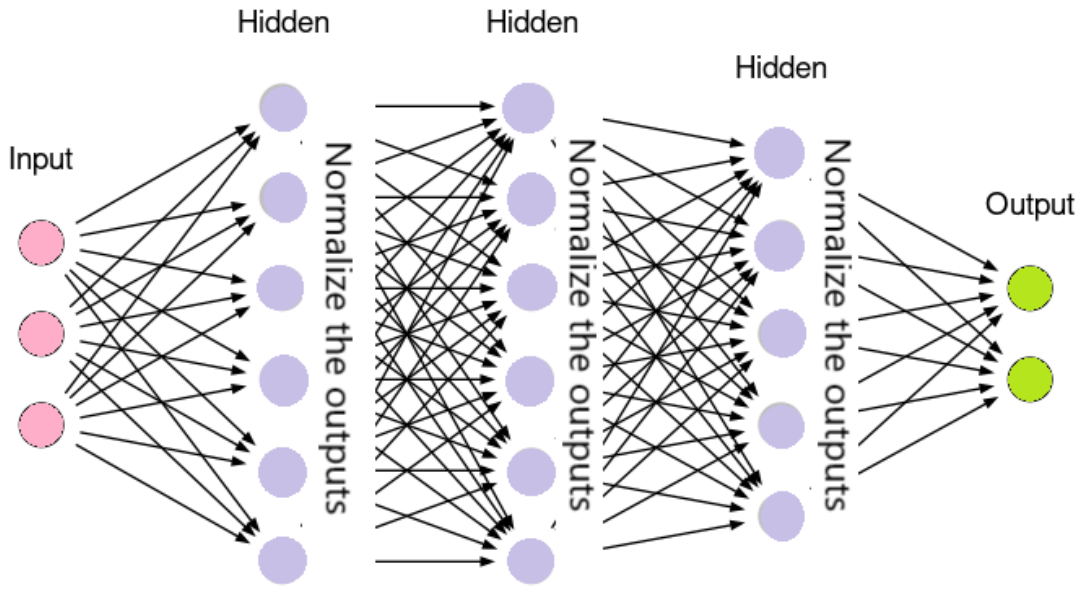


Fig. 4.5.1: Batch normalization layers [21]

4.6 Masking

Masking (M) informs sequence-processing layers that specific timesteps in the input are missing and should be skipped when processing the data [5]. If all values in the input tensor at that timestep are equal to the mask value, the timestep will be

masked in all downstream layers. For our experiments, we used TensorFlow Keras Masking [5] to implement this countermeasure. A general visualization of masking is shown in an MNIST example in Figure 4.6.1. Our work is the first to utilize it as a protection against MIA in both CL and FL setting.

Fig. 4.6.1: Masking on MNIST images [50]

4.7 Activity Regularization

Activity Regularization (AR) is a neural network method to learn the data's internal representations. L1 regularizer and L2 regularizer are the two regularisation techniques [3]. L1 regularization penalizes the sum of absolute values of the weights, whereas L2 regularization penalizes the sum of squares of the weights. The L1 and L2 have stated as the L1 norm allows some weights to be significant while driving others to zero. It penalizes the actual value of a weight. The L2 norm causes all weights to decrease in size. It penalizes the square value of a weight. For our experiments, we used TensorFlow Keras Activity regularization [6] to implement this countermeasure and set the L1 and L2 regularization factors at 0.01. Shokri et al. [46] used the conventional L2-regularizer as a defense technique in the CL setting to overcome the MIA on ML models trained using a neural network.

4.8 Differential Privacy

Differential Privacy (DP) is a framework for estimating the privacy provided by an algorithm [8]. Through DP, one can design machine learning algorithms that responsibly train models on private data. Learning with DP guarantees privacy, mitigating

the risk of exposing sensitive training data in machine learning. Single or small training examples should not influence a model trained with DP in its dataset, which helps mitigate the risk of exposing sensitive training data in ML. The basic idea of Differentially Private Stochastic Gradient Descent (DP-SGD) is to modify the gradients used in Stochastic Gradient Descent (SGD), which is the core of almost all deep learning algorithms. DP-SGD algorithm incorporates Gaussian additive noise with the gradient updates in SGD to ensure DP during model training [12]. Models trained with DP-SGD provide DP guarantees for their input data. There are two changes made to the SGD algorithm:

- First, the sensitivity of each gradient needs to be determined. In other words, there must be a limit to how much each training point sampled in a minibatch can affect gradient computations and the resulting updates applied to model parameters, which can be done by clipping each gradient computed on each training point.

- Random noise is sampled and added to the clipped gradients to make it statistically impossible to know whether or not a particular data point was included in the training dataset by comparing the updates SGD applies when it operates with or without this particular data point in the training dataset. TensorFlow Privacy delivers code that covers an existing TensorFlow optimizer to create a variant that implements DP-SGD.

DP [26, 25, 27] is a strong standard for ensuring privacy in distributed datasets. For our experiments, we used TensorFlow Keras Differential privacy [8] to implement this countermeasure.

4.9 Knowledge Distillation

Knowledge Distillation (KD) distills and transfers knowledge from one deep neural network to another deep neural network [55]. It is a method to compress the model while maintaining accuracy. The larger network that gives the knowledge is called a

Teacher network, and the smaller network that receives the knowledge is called a Student network. The teacher network is first trained separately until full convergence. Afterward, the distiller then transfers knowledge from the teacher to the student. The student network is trained in coordination with the fully trained teacher network.

A generic illustration of knowledge distillation is shown in Figure 4.9.1. According to many works in ML against MIA, KD outperforms other mitigating approaches [56, 45], while other works in FL claim that it facilitates effective communication [51, 30, 33]. Our experiments used Keras knowledge distillation [4] to implement this countermeasure.

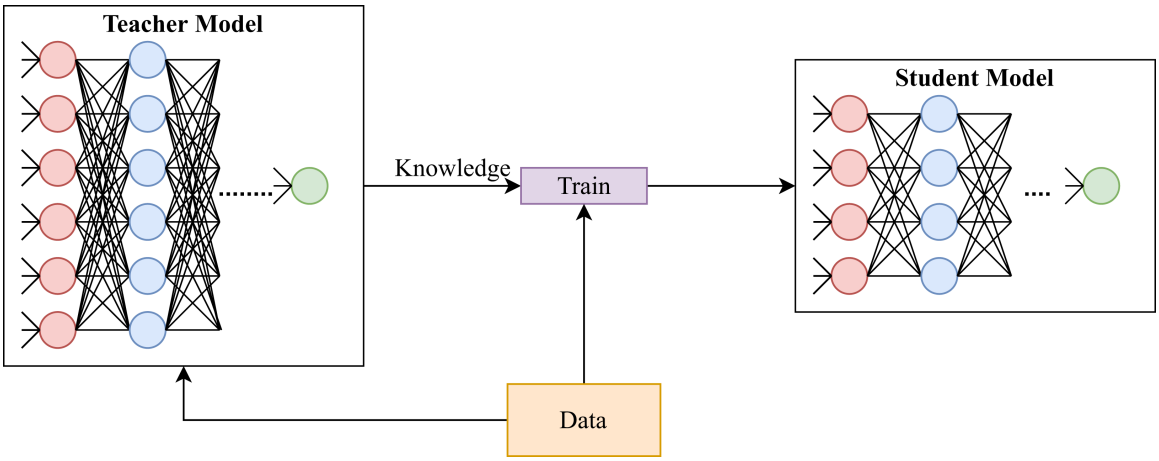


Fig. 4.9.1: A generic illustration of knowledge distillation [13]

4.10 Conclusion

In this chapter, we discussed various countermeasures in detail and discussed whether they were used in earlier work. Countermeasures such as Dropout, Gaussian Noise, and Activity Regularization are already used to mitigate membership inference attacks in a centralized learning setting, we implemented them in a Federated learning setting as well. Batch Normalization, Differential Privacy, and Knowledge Distillation are used as a countermeasure in the federated learning environment. Our work is the first to utilize Gaussian Dropout, Monte Carlo Dropout, and Masking as a countermeasure against MIA in both CL and FL settings.

CHAPTER 5

Performance Analysis

In this chapter, we described the experimental setup and results. In the experimental setup, we discussed the datasets and dataset preprocessing. Then we discussed the architecture of our model and its training settings. We also discussed the comparison techniques and evaluation metrics we applied for our experiments. Next, we discuss the experimental results for CL and FL with and without countermeasures.

5.1 Experimental Setup

This section explained the datasets we used, the model architecture, and its training parameters. We also detailed evaluation metrics and comparison techniques to conduct various MIA attacks in the FL environment.

5.1.1 Datasets

We conducted experiments on three datasets MNIST [22], FMNIST [52], and CIFAR-10 [31] to evaluate the attacks in the FL setting. Table 5.1.1 shows general information about all the datasets we used. These datasets are the benchmark to validate MIA and are the same as those used in related work. Figure 5.1.1 and Figure 5.1.2 show grayscale visualization of MNIST [22] and FMNIST [52], respectively whereas Figure 5.1.3 shows RGB visualization of CIFAR-10 [31].

- MNIST is a freely accessible dataset that contains 70,000 images of handwritten digits, 60,000 images of the training set, and 10,000 images of the testing set. Each image is formatted as 28 x 28 and processed so that the digit is in

Fig. 5.1.1: Visualization of MNIST Dataset [22]

the center. The MNIST dataset is a 10-class classification problem in which the task is to determine which digit between 0 and 9, inclusive, is present in a given image.

- FMNIST is a dataset that consists of 70,000 images of Zalando's article images, 60,000 images of the training set, and 10,000 images of the testing set. Each image is a 28 x 28 grayscale image associated with a label from 10 classes.
- CIFAR-10 dataset is also freely accessible and contains 60,000 color images,

Fig. 5.1.2: Visualization of FMNIST Dataset [52]

50,000 images of the training set, and 10,000 images of the test set. Each image is again formatted to be 32 x 32. There are also ten classes in the CIFAR-10 dataset: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class has 6,000 images available. The problem is a 10-class classification problem to determine which of the ten classes is depicted in a given image.

Fig. 5.1.3: Visualization of CIFAR-10 Dataset [31]

Table 5.1.1: General information of the datasets

Name	Type	Number of Classes	Number of Training Records	Number of Testing Records
MNIST	Image	10	60,000	10,000
FMNIST	Image	10	60,000	10,000
CIFAR-10	Image	10	50,000	10,000

5.1.2 Dataset Preprocessing

We divided each of the datasets into 30,000 for training and 10,000 for testing to set up the FL environment, as shown in Table 5.1.2. Then the dataset is divided into

each participant 10,000 for training and 1,000 for teasing, to train separately using the FedAvg [35] algorithm, as shown in Table 5.1.3.

Table 5.1.2: Size of the dataset to train and test in the FL environment

Datasets	**Dataset for FL Setting**	
	Training Dataset	**Testing Dataset**
MNIST	30,000	10,000
FMNIST	30,000	10,000
CIFAR-10	30,000	10,000

We normalized the image data to the range [0,1], which allows the model to train more efficiently.

Table 5.1.3: Number of participants and size of the dataset to train and test participants in the FL environment

Number of Participants	**Participants dataset for FL setting**	
	Training Dataset	**Testing Dataset**
Participant 1	10,000	1,000
Participant 2	10,000	1,000
Participant 3	10,000	1,000

5.1.3 Model Architecture

We performed our experiments on 1.90GHz Intel(R) Core(TM) i3-4030U processor with 4.00 GB RAM on the x64-based processor. We used open-source frameworks and standard libraries, such as Keras and Tensorflow, that can be deployed on endpoints.

We created a model using Keras' Sequential function Object and then added a linear stack of network layers. Our model first defines the flattened input layer, followed by three Dense layers. MNIST and FMNIST input sizes are 28,28,1, while CIFAR-10 input sizes are 32,32,3. Between the dense layers, we added countermeasure

layers as shown in Figure 5.1.4. We used the countermeasure layers as our experiment demanded. We specify an output size of 10 and set the activation function for this layer to softmax because we are using our network to solve a multi-class classification problem.

5.1.4 Training Setting

For the learning process, we used three optimizers: SGD, RMSProp, and Adagrad, with a learning rate of 0.01. We used categorical cross-entropy loss for the loss function and included the accuracy metric, with a batch size of 32 and epochs of 10, for training the participants' models. The FL process is replicated, including local participant training and client-server aggregation. We used the FedAvg [35] algorithm to update the FL global model with three participants, as illustrated in Figure 5.1.5.

5.1.5 Evaluation Metrics

We used the test accuracy as an evaluation metric for the FL model. The standard metric for measuring attack accuracy is recall. We used recall as an evaluation metric for comparing MIA attacks in the FL environment. The recall represents the fraction of the training dataset members correctly inferred as members by the attacker.

5.1.6 Comparison Methods

We compared four attacks as shown in Table 5.1.4. Attack 1 uses multiple shadow models with the same structure and training data distribution as the target model. Attack 2 uses a single shadow model; its structure is different, and training data distribution is the same as the target model. In contrast, Attack 3 training data distribution differs from the target model and uses a single shadow model. Lastly, Attack 4 uses the Jacobian matrix and prediction sensitivity technique.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
mc_dropout (MCDropout)       (None, 784)               0
_________________________________________________________________
gaussian_noise (GaussianNois (None, 784)               0
_________________________________________________________________
gaussian_dropout (GaussianDr (None, 784)               0
_________________________________________________________________
batch_normalization (BatchNo (None, 784)               3136
_________________________________________________________________
activity_regularization (Act (None, 784)               0
_________________________________________________________________
masking (Masking)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 300)               235500
_________________________________________________________________
mc_dropout_1 (MCDropout)     (None, 300)               0
_________________________________________________________________
gaussian_noise_1 (GaussianNo (None, 300)               0
_________________________________________________________________
gaussian_dropout_1 (Gaussian (None, 300)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 300)               1200
_________________________________________________________________
activity_regularization_1 (A (None, 300)               0
_________________________________________________________________
masking_1 (Masking)          (None, 300)               0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               30100
_________________________________________________________________
mc_dropout_2 (MCDropout)     (None, 100)               0
_________________________________________________________________
gaussian_noise_2 (GaussianNo (None, 100)               0
_________________________________________________________________
gaussian_dropout_2 (Gaussian (None, 100)               0
_________________________________________________________________
batch_normalization_2 (Batch (None, 100)               400
_________________________________________________________________
activity_regularization_2 (A (None, 100)               0
_________________________________________________________________
masking_2 (Masking)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1010
=================================================================
Total params: 271,346
Trainable params: 268,978
Non-trainable params: 2,368
```

Fig. 5.1.4: Visualization of model architecture

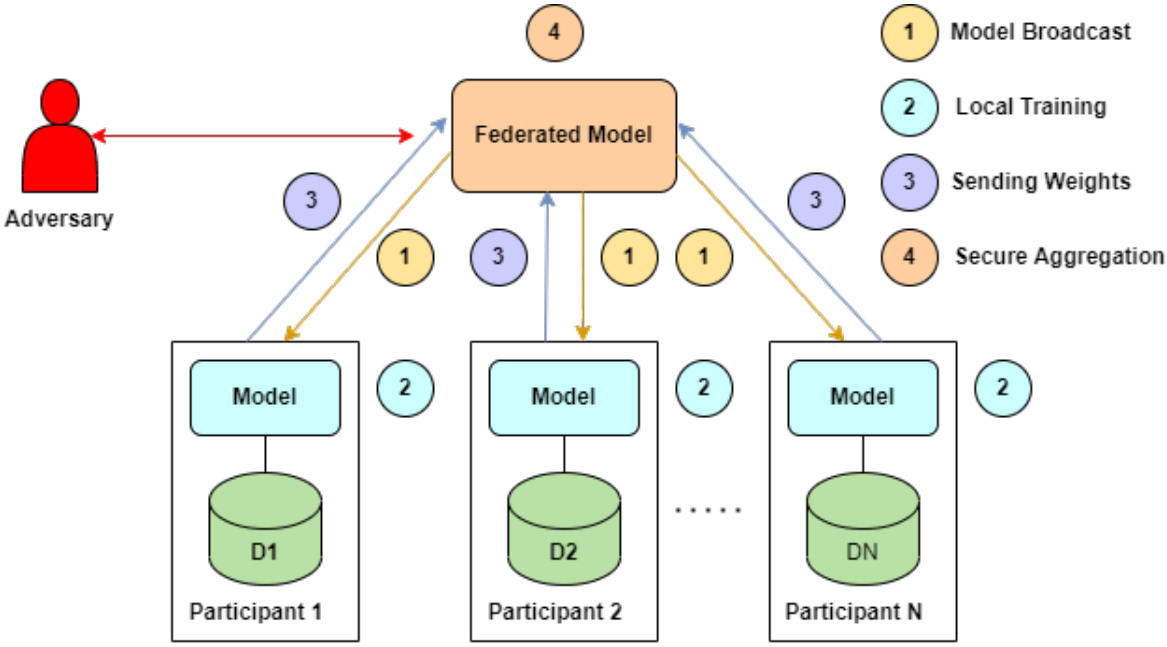


Fig. 5.1.5: Overview of a FL system

Table 5.1.4: General information of the attacks

Attack type	Shadow model		Target's model training data distribution	Prediction sensitivity
	No. of. shadow models	Target model structure		
Attack 1 [46]	multiple	✓	✓	-
Attack 2 [44]	1	-	✓	-
Attack 3 [44]	1	-	-	-
Attack 4 [34]	-	-	-	✓

5.2 Experimental Results

In this section, we presented our experimental results. We started by comparing CL and FL model accuracy with their attack recall. Then we presented FL model accuracy and compared all the MIA attacks in the FL environment.

5.2.1 CL vs FL

Many studies thoroughly compared CL and FL approaches [15, 42]. FL is concluded as a network-efficient alternative to CL [24]. In our comparison of the two approaches, as shown in Figure 5.2.1, CL outperformed FL regarding accuracy, which is expected.

Generally, the recall of all Attacks 1, 2, 3, and 4 is almost the same, if not less, in FL compared to the recall in CL, considering different mitigation techniques. Comparison of CL and FL attacks using SGD optimizer on MNIST dataset is shown in Figure 5.2.2, Figure 5.2.3, Figure 5.2.4 and Figure 5.2.5.

We achieved the highest CL model accuracy(92.8) using BN with SGD on the MNIST dataset and the lowest CL model accuracy(72.6) using MCD with Adagrad on the CIFAR-10 dataset with countermeasures, as shown in Table 5.2.1. Generally, for each optimizer, CL model accuracy is the lowest for CIFAR-10 and the highest for MNIST, which is the same as in the FL case.

As shown in Table 5.2.1, BN does not affect the MNIST model accuracy. For FMNIST and CIFAR-10, the countermeasure that does not affect the model accuracy depends on the considered optimizer.

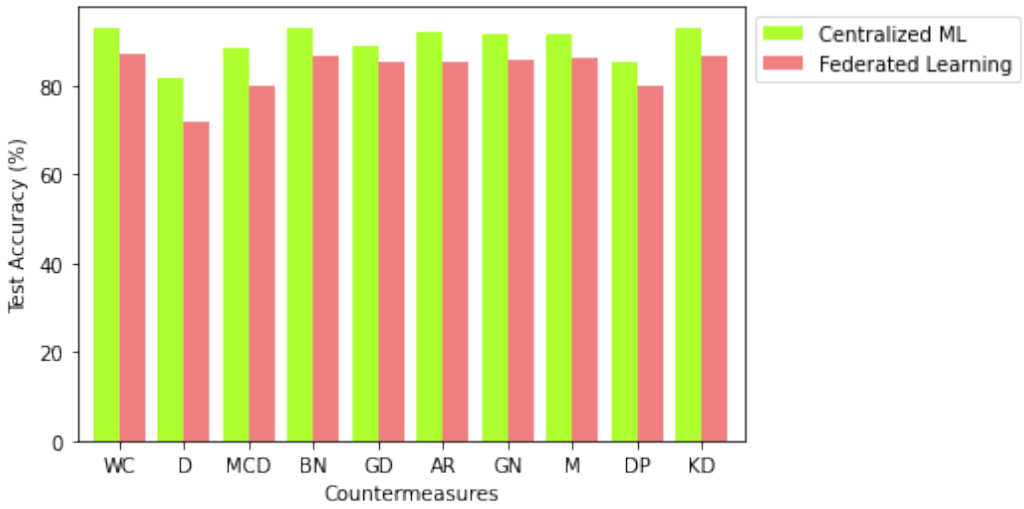


Fig. 5.2.1: CL vs FL model accuracy on SGD optimizer - MNIST

The CL attack recall for all the optimizers, countermeasures, and datasets considering the four attacks is shown in Table 5.2.2. It is observed that countermeasures on Attack 4 decrease the attack recall significantly. Generally, KD and MCD are considered the best mitigation on CL settings with all datasets because they decrease the attack recall more than other countermeasures.

Table 5.2.1: CL model accuracy

Datasets	Optimizers	WC	D	MCD	BN	GD	AR	GN	M	KD	DPSGD
MNIST	SGD	93.1	81.6(-11.5)	88.3(-4.8)	**92.8(-0.3)**	89.1(-4)	92.1(-1)	91.4(-1.7)	91.6(-1.5)	**92.8(-0.3)**	85.5(-7.6)
	Adagrad	92.6	86.5(-6.1)	85.7(-6.9)	**92.1(-0.5)**	89.6(-3)	91.6(-1)	91.1(-1.5)	90.3(-2.3)	92(-0.6)	-
	RMSProp	92.8	92.5(-0.3)	90.9(-1.9)	**92.7(-0.1)**	91.3(-1.5)	90.4(-2.4)	90.7(-2.1)	91.5(-1.3)	91.2(-1.6)	-
FMNIST	SGD	88.6	84.7(-3.9)	84.1(-4.5)	86.4(-2.2)	85.2(-3.3)	87.3(-1.3)	85.8(-2.8)	**88.1(-0.5)**	86.9(-1.7)	84.2(-4.4)
	Adagrad	85.8	78.6(-7.2)	75.9(-9.9)	**84.6(-1.2)**	81.9(-3.9)	83.8(-2)	79.1(-6.7)	88.7(+2.9)	82.3(-3.5)	-
	RMSProp	83.6	79.5(-4.1)	78.5(-5.1)	82.9(-0.7)	78.7(-4.9)	82.6(-1)	81.7(-1.9)	**83.1(-0.5)**	81.9(-1.7)	-
CIFAR-10	SGD	85.7	74.2(-11.5)	74.9(-10.8)	82.8(-2.9)	83.1(-2.6)	**84.3(-1.4)**	78.6(-7.1)	82.5(-3.2)	83.9(-1.8)	74.6(-11.1)
	Adagrad	88.4	75.7(-12.7)	72.6(-15.8)	85.8(-2.6)	83.6(-4.8)	**87.2(-1.2)**	81.9(-6.5)	82.5(-5.9)	85.3(-3.1)	-
	RMSProp	86.3	81.6(-4.7)	76.4(-9.9)	**85.7(-0.6)**	84.9(-1.4)	84.2(-2.1)	82.1(-4.2)	83.6(-2.7)	81.5(-4.8)	-

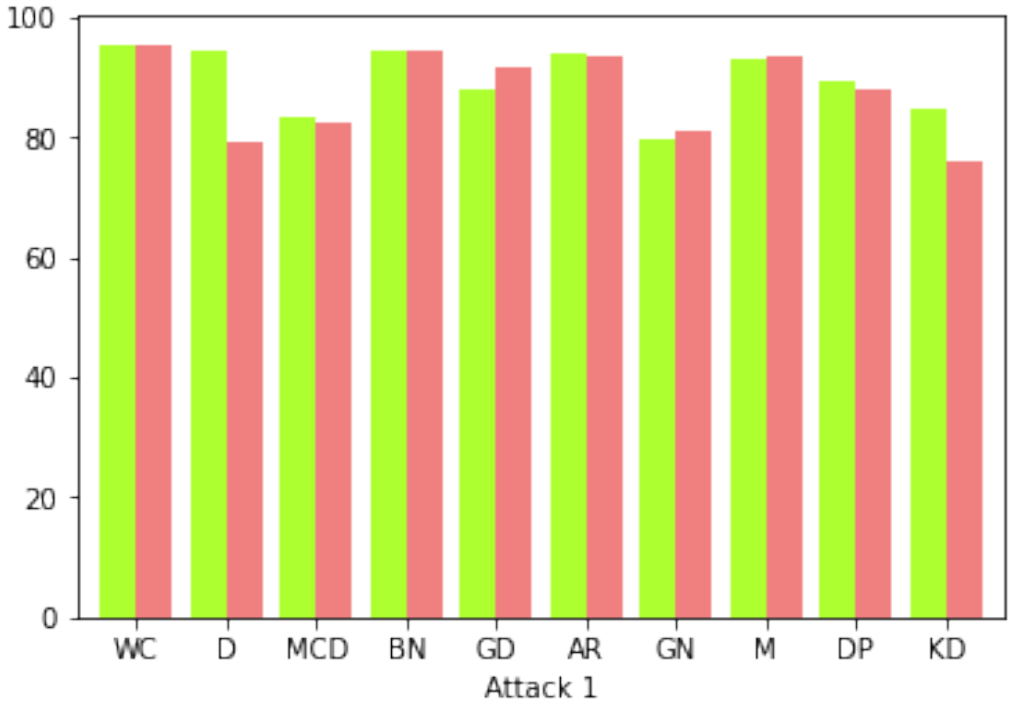


Fig. 5.2.2: CL vs FL Attack 1 recall on SGD optimizer - MNIST

5.2.2 Analysis of FL Model Accuracy

The effects of various optimizers and countermeasures on FL model accuracy on the three datasets are shown in Figure 5.2.6, Figure 5.2.7, and Figure 5.2.8. The y-axis represents the test accuracy of the FL model, and the x-axis represents various countermeasures. We used DP-SGD to perform DP only on the SDG optimizer. The first bar in all the plots represents the accuracy without countermeasures.

Table 5.2.2: CL Attack Recall

Datasets	Optimizers	Attacks	WC	D	MCD	BN	GD	AR	GN	M	KD	DPSGD
MNIST	SGD	Attack-1	95.4	94.2(-1.2)	83.4(-12)	94.4(-1)	87.9(-7.5)	94(-1.4)	**79.8(-15.6)**	93.1(-2.3)	84.6(-10.8)	89.2(-6.2)
		Attack-2	94.9	93.7(-1.2)	**83.2(-11.7)**	94.8(-0.1)	86.8(-8.1)	93.6(-1.3)	93.2(-1.7)	93.9(-1)	82.4(-12.5)	88.3(-6.6)
		Attack-3	90.7	85.3(-5.4)	**74.5(-16.2)**	88.7(-2)	82.9(-7.8)	83.2(-7.5)	88.6(-2.1)	89.3(-1.4)	80.5(-10.2)	82.4(-8.3)
		Attack-4	87.1	32(-55.1)	34.6(-52.5)	28.7(-58.4)	24.5(-62.2)	35.3(-51.8)	37.4(-49.7)	24.6(-62.5)	**22.6(-64.5)**	26.8(-60.3)
	Adagrad	Attack-1	97.8	97.2(-0.6)	93.8(-4)	96.6(-1.2)	95.2(-2.6)	96.9(-0.9)	95.4(-2.4)	96.9(-0.9)	**94.6(-3.2)**	-
		Attack-2	97.7	92.4(-5.3)	93.5(-4.2)	95.7(-2)	95.1(-2.6)	96.4(-1.3)	93.6(-4.1)	95.9(-1.8)	**76.1(-21.6)**	-
		Attack-3	91.3	87.4(-3.9)	76.6(-14.7)	89.5(-1.8)	86.2(-5.1)	86.3(-5)	81.3(-10)	85.4(-5.9)	**74.9(-16.4)**	-
		Attack-4	86.9	33.6(-53.3)	32.1(-54.8)	35.2(-51.7)	34.1(-52.8)	39.7(-47.2)	31.8(-55.1)	35.3(-51.6)	**31.4(-55.5)**	-
	RMSProp	Attack-1	99.4	98.9(-0.5)	98.3(-1.1)	99.3(-0.1)	98.8(-0.6)	99.1(-0.3)	98.6(-0.8)	98.4(-1)	**98.2(-1.2)**	-
		Attack-2	99.2	98.6(-0.6)	97.3(-1.9)	98.7(-0.5)	97.2(-2)	98.3(-0.9)	97.7(-1.5)	97(-2.2)	**96.4(-2.8)**	-
		Attack-3	98.6	96.3(-2.3)	94.1(-4.5)	98.2(-0.4)	93.1(-5.5)	95.6(-3)	94.3(-4.3)	94.8(-3.8)	**92.5(-6.1)**	-
		Attack-4	89.9	23(-66.9)	38.7(-51.2)	39.5(-50.4)	37.3(-52.6)	39.4(-50.4)	34.6(-55.3)	32.8(-57.1)	**31.3(-58.6)**	-
FMNIST	SGD	Attack-1	95.8	93.7(-2.1)	85.6(-10.2)	95.3(-0.5)	88.9(-6.9)	94.7(-1.1)	86.5(-9.3)	93.4(-2.4)	**83.5(-12.3)**	86.3(-9.5)
		Attack-2	93.6	86.4(-7.2)	83.2(-10.4)	93.1(-0.5)	85.9(-7.7)	92.8(-0.8)	86.1(-7.5)	92.5(-1.1)	**82.9(-10.7)**	85.7(-7.9)
		Attack-3	90.2	82.2(-8)	**81.6(-8.6)**	89.6(-0.6)	84.9(-5.3)	89.1(-1.1)	83.7(-6.5)	88.5(-1.7)	81.9(-8.3)	83.1(-7.1)
		Attack-4	82.1	27(-55.1)	**25.8(-56.3)**	33.6(-48.5)	38.4(-43.7)	42.8(-39.3)	29.5(-52.6)	37.5(-44.6)	21.7(-60.4)	27.9(-54.2)
	Adagrad	Attack-1	90.6	83.2(-7.4)	**82.4(-8.2)**	85.4(-5.2)	83.6(-7)	86.9(-3.7)	87.6(-3)	89.5(-1.1)	84.2(-6.2)	-
		Attack-2	87.2	82.6(-4.6)	**81.6(-5.6)**	84.3(-2.9)	81.9(-5.3)	85.3(-1.9)	86.8(-0.4)	85.7(-1.5)	81.8(-5.4)	-
		Attack-3	85.7	82.4(-3.3)	81.1(-4.6)	82.8(-2.9)	**78.4(-7.3)**	82.9(-2.8)	82.7(-3)	83.6(-2.1)	79.3(-6.4)	-
		Attack-4	80.9	46.2(-34.7)	26.9(-54)	36.7(-44.2)	**26.4(-54.5)**	35.4(-45.5)	32(-48.9)	43.2(-37.7)	31.2(-49.7)	-
	RMSProp	Attack-1	91.5	76.4(-15.1)	73.8(-17.7)	89.4(-2.1)	86.9(-4.6)	88.4(-3.1)	85.2(-6.1)	87.5(-4)	**72.6(-18.9)**	-
		Attack-2	89.2	73.6(-15.6)	72.7(-16.5)	86.3(-2.9)	82.8(-6.4)	87.2(-2)	83.1(-6.1)	85.3(-3.9)	**71.8(-17.4)**	-
		Attack-3	85.3	71.9(-13.4)	70.6(-14.7)	84.3(-1)	81.4(-3.9)	82.8(-2.5)	80.1(-5.2)	81.6(-3.7)	**69.3(-16)**	-
		Attack-4	70.8	**20.1(-50.7)**	20.5(-50.3)	35.9(-34.9)	26.2(-44.6)	34.8(-36)	28.6(-42.2)	29.6(-41.2)	26.1(-44.7)	-
CIFAR-10	SGD	Attack-1	92.6	82.8(-9.8)	**80.5(-12.1)**	91.7(-0.9)	91.5(-1.1)	89.4(-3.2)	85.3(-7.3)	90.4(-2.2)	81.3(-11.3)	84.2(-8.4)
		Attack-2	90.4	80.2(-10.2)	79.9(-10.5)	89.2(-1.2)	89.6(-0.8)	84.2(-6.2)	83.7(-6.7)	86.9(-3.5)	**79.4(-11)**	81.8(-8.6)
		Attack-3	84.7	77.9(-6.8)	75.1(-9.6)	82.8(-1.9)	82.6(-2.1)	81.7(-3)	82.3(-2.4)	82.8(-1.9)	**75.2(-9.5)**	76.4(-8.3)
		Attack-4	78.4	36.8(-41.6)	35.6(-42.8)	42.7(-35.7)	40.9(-37.5)	40.5(-37.9)	39.1(-39.3)	40.6(-37.8)	**33.9(-44.5)**	35.2(-43.2)
	Adagrad	Attack-1	91.3	76.8(-14.5)	75.7(-15.6)	90.2(-1.1)	89.4(-1.9)	90.4(-0.9)	87.6(-3.7)	88.1(-3.3)	**74.2(-17.1)**	-
		Attack-2	89.4	73.6(-15.8)	**72.4(-17)**	88.3(-1.1)	87.9(-1.5)	85.3(-4.1)	84.1(-5.3)	83.8(-5.6)	72.8(-16.6)	-
		Attack-3	83.6	68.9(-14.7)	**65.7(-17.9)**	83.1(-0.5)	80.7(-2.9)	81.5(-2.1)	81.6(-2)	82.4(-1.2)	66.9(-16.7)	-
		Attack-4	72.5	28.6(-43.9)	25.3(-47.2)	36.4(-36.1)	34.9(-37.6)	37.7(-34.8)	30.3(-42.2)	33.4(-39.1)	**25.9(-46.6)**	-
	RMSProp	Attack-1	89.3	83.6(-5.7)	82.9(-6.4)	88.4(-0.9)	87.3(-2)	88(-1.3)	87.6(-1.7)	89.1(-0.2)	**82.5(-6.8)**	-
		Attack-2	84.2	78.4(-5.8)	**75.9(-8.3)**	83.9(-0.3)	82.4(-1.8)	81.9(-2.3)	82.8(-1.4)	82.7(-1.5)	76.8(-7.4)	-
		Attack-3	81.6	74.9(-6.7)	72.9(-8.7)	80.6(-1)	78.5(-3.1)	77.3(-4.3)	78.2(-3.4)	80.1(-1.5)	**71.4(-10.2)**	-
		Attack-4	68.5	24.7(-43.8)	**22.6(-45.9)**	32.9(-35.6)	30.5(-38)	31.8(-36.7)	29.2(-39.3)	31.3(-37.2)	23.4(-45.1)	-

5.2.2.1 FL model accuracy without countermeasure

We achieved the highest FL model accuracy using RMSProp on the MNIST dataset and the lowest FL model accuracy using RMSProp on the CIFAR-10 dataset as shown in Table 5.2.3. Generally, for each optimizer, FL model accuracy is the lowest for CIFAR-10 and the highest for MNIST. This can be justified by the nature of the dataset.

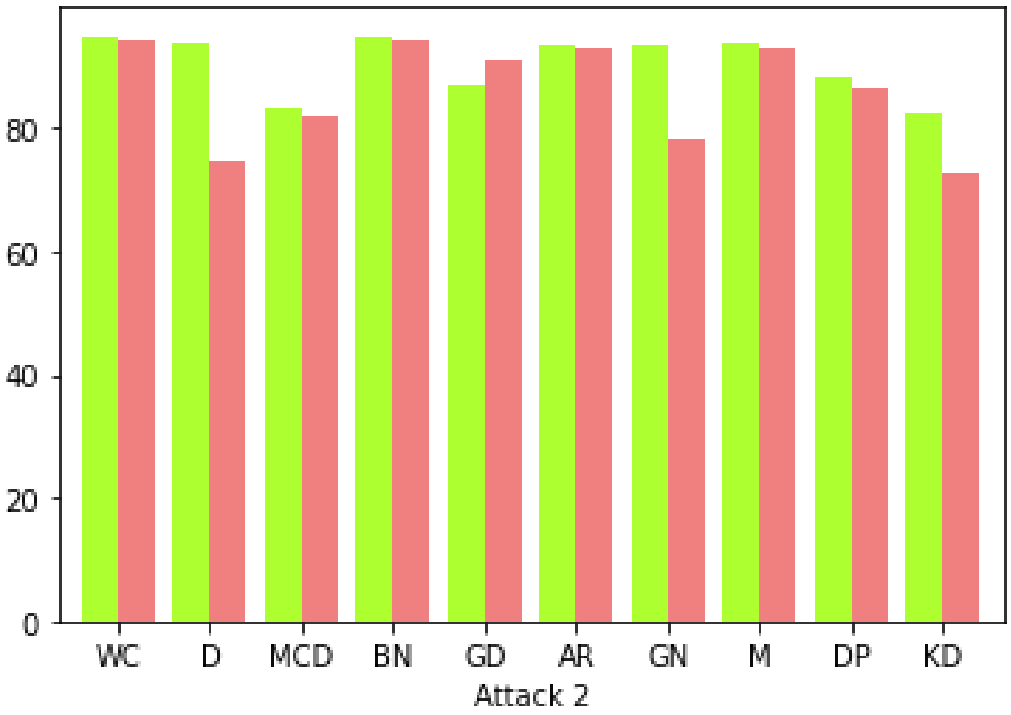


Fig. 5.2.3: CL vs FL Attack 2 recall on SGD optimizer - MNIST

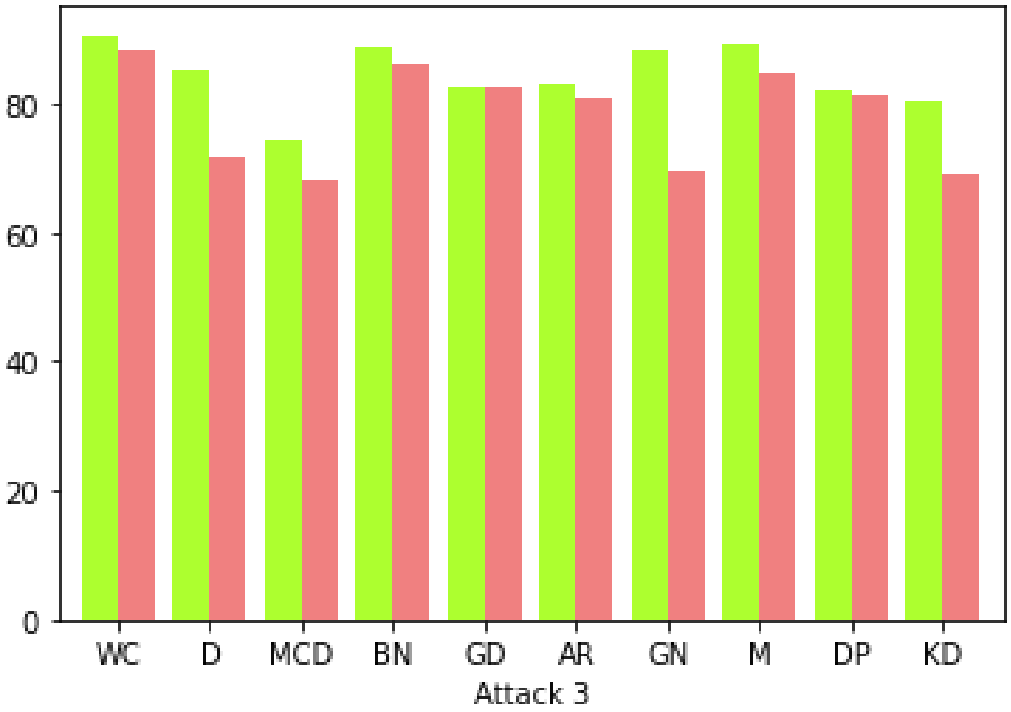


Fig. 5.2.4: CL vs FL Attack 3 recall on SGD optimizer - MNIST

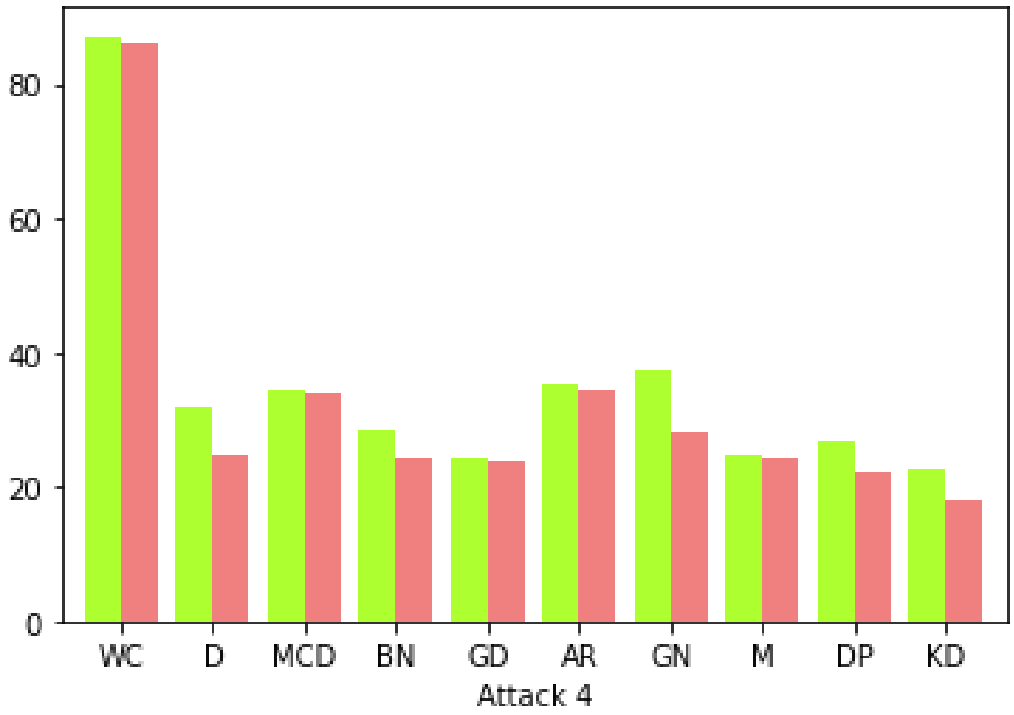


Fig. 5.2.5: CL vs FL Attack 4 recall on SGD optimizer - MNIST

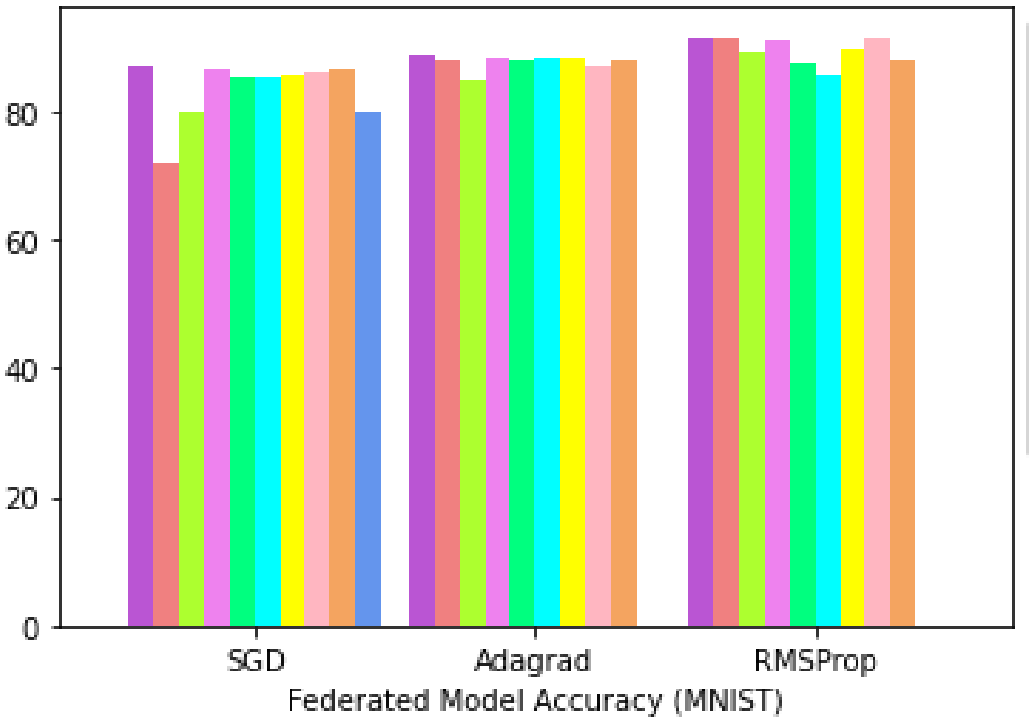


Fig. 5.2.6: FL model accuracy on all optimizer - MNIST

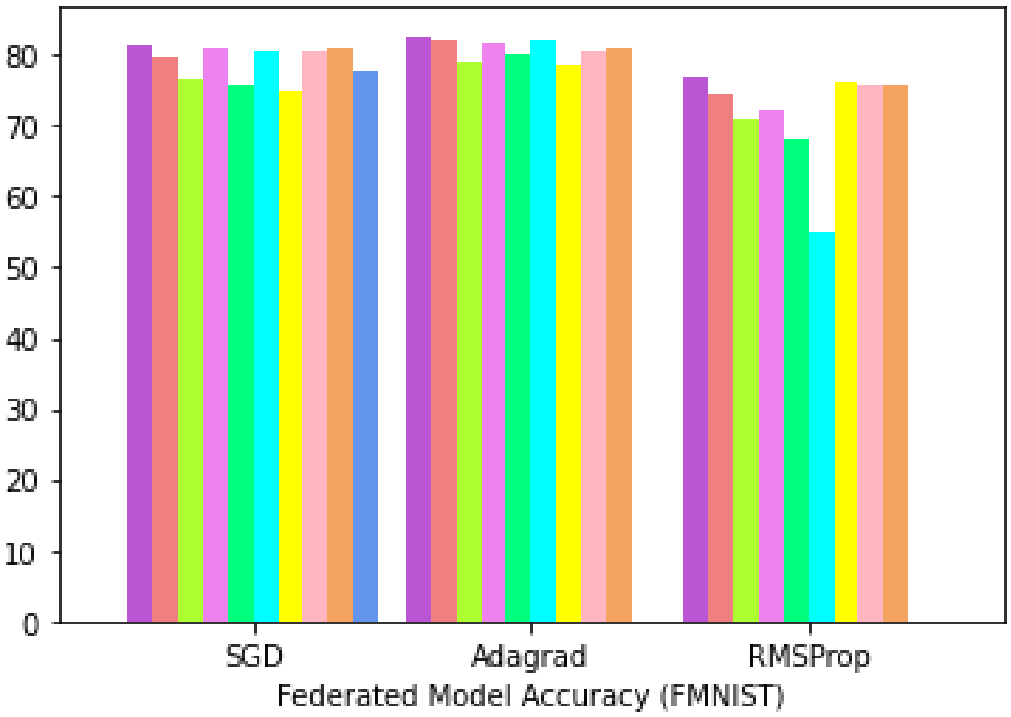


Fig. 5.2.7: FL model accuracy on all optimizer - FMNIST

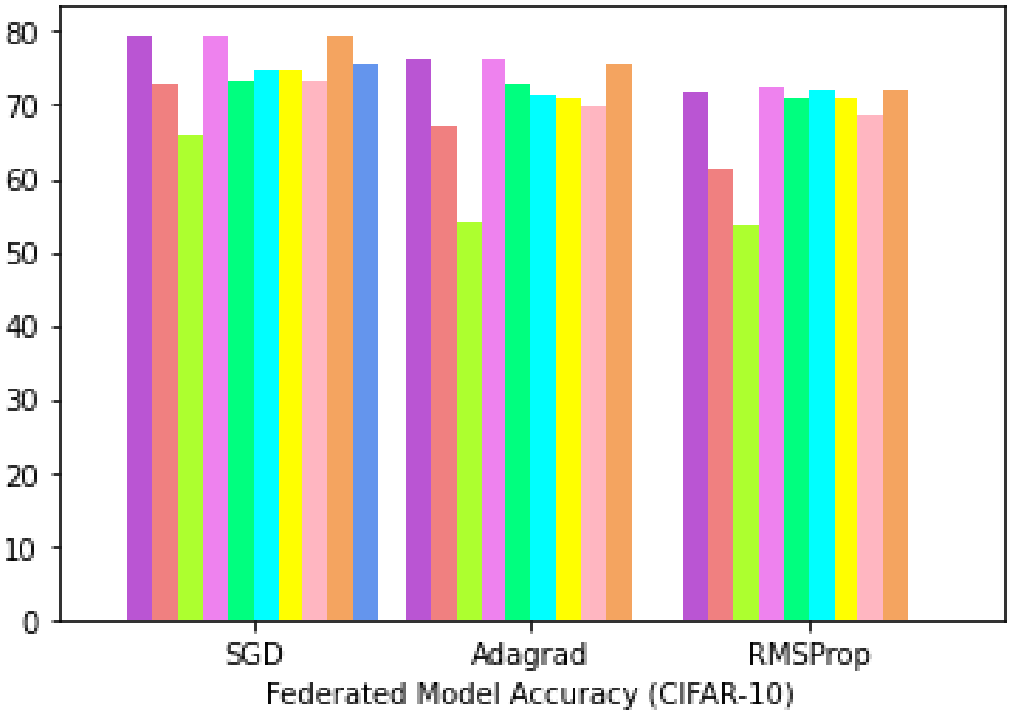
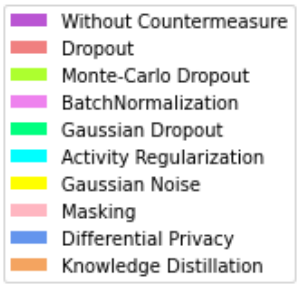


Fig. 5.2.8: FL model accuracy on all optimizer - CIFAR-10



	Countermeasures
WC	Without Countermeasure
D	Dropout
MCD	Monte Carlo Dropout
BN	Batch Normalization
GD	Gaussian Dropout
AR	Activity Regularization
GN	Gaussian Noise
M	Masking
DPSGD	Differential Privacy-SGD
KD	Knowledge Distillation

Table 5.2.3: FL model accuracy

Datasets	**Optimizers**	**WC**	**D**	**MCD**	**BN**	**GD**	**AR**	**GN**	**M**	**KD**	**DPSGD**
MNIST	SGD	87	72(-15)	80(-7)	86.7(-0.3)	85.5(-1.5)	85.4(-1.6)	85.6(-1.4)	86.2(-0.8)	**86.9(-0.1)**	79.9(-7.1)
	Adagrad	88.7	88.2(-0.5)	84.9(-3.8)	88.3(-0.4)	88.1(-0.6)	**88.5(-0.2)**	88.3(-0.4)	87(-1.7)	88.2(-0.5)	-
	RMSProp	91.7	91.6(-0.1)	89.5(-2.2)	91.1(-0.6)	87.4(-4.3)	86(-5.7)	90(-1.7)	**91.7(0)**	87.9(-3.8)	-
FMNIST	SGD	81.3	79.8(-1.5)	76.4(-4.9)	**81(-0.3)**	75.7(-5.6)	80.5(-0.8)	74.9(-6.4)	80.3(-1)	80.9(-0.4)	77.6(-3.7)
	Adagrad	82.6	82(-0.6)	78.9(-3.7)	81.6(-1)	79.9(-2.7)	**82.2(-0.4)**	78.4(-4.2)	80.6(-2)	80.7(-1.9)	-
	RMSProp	91.7	**76.8(-14.9)**	74.4(-17.3)	71(-20.7)	72.3(-19.4)	68.2(-23.5)	55(-36.7)	76.1(-15.6)	75.8(-15.9)	-
CIFAR-10	SGD	79.5	73(-6.5)	65.9(-13.6)	**79.3(-0.2)**	73.2(-6.3)	74.6(-4.9)	74.9(-4.6)	73.1(-6.4)	**79.3(-0.2)**	75.7(-3.8)
	Adagrad	76.3	67(-9.3)	54(-22.3)	**76.2(-0.1)**	72.9(-3.4)	71.4(-4.9)	71.1(-5.2)	69.9(-6.4)	75.7(-0.6)	-
	RMSProp	72.8	61.2(-11.6)	53.6(-19.2)	**72.4(-0.4)**	70.9(-1.9)	72.2(-0.6)	71(-1.8)	68.6(-4.2)	72.1(-0.7)	-

5.2.2.2 FL model accuracy with countermeasures

As shown in Table 5.2.3, BN does not affect the CIFAR-10 model accuracy. For MNIST and FMNIST, the countermeasure that does not affect the model accuracy depends on the considered optimizer.

5.2.3 Analysis of the Attack Recall

Reducing the attacks' recall is the best way to mitigate the MIA. Figure 5.2.9, Figure 5.2.10, Figure 5.2.11, Figure 5.2.12, Figure 5.2.13, Figure 5.2.14, Figure 5.2.15, Figure 5.2.16 and Figure 5.2.17 shows the results of the four attacks using three optimizers, with and without countermeasures on three datasets, respectively. The y-axis represents the recall of the attack, and the x-axis represents various countermeasures on a particular optimizer.

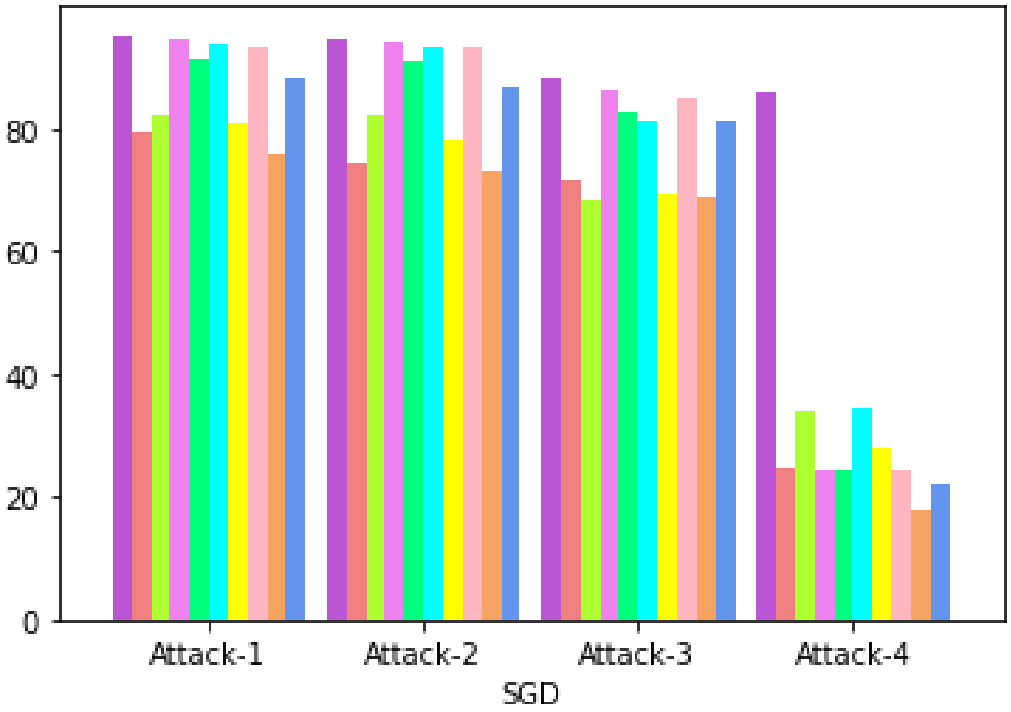


Fig. 5.2.9: A comparison of the four attacks using SGD optimizers with and without countermeasures - MNIST

5.2.3.1 Attacks without countermeasure

Attack 4 has the lowest attack recall with MNIST. Attack 4 and Attack 3 have the lowest attack recall with FMNIST and CIFAR-10. On the other hand, Attack 1 has the highest attack recall for all datasets, as shown in Table 5.2.4.

Table 5.2.4: FL Attack Recall

Datasets	Optimizers	Attacks	WC	D	MCD	BN	GD	AR	GN	M	KD	DPSGD
MNIST	SGD	Attack-1	95.2	79.3(-15.9)	82.4(-12.8)	94.5(-0.7)	91.6(-3.6)	93.6(-1.6)	81(-14.2)	93.4(-1.8)	**75.7(-19.5)**	88(-7.2)
		Attack-2	94.5	74.6(-19.9)	82(-12.5)	94.3(-0.2)	91(-3.5)	93.1(-1.4)	78.2(-16.3)	93(-1.5)	**72.8(-21.7)**	86.7(-7.8)
		Attack-3	88.2	71.7(-16.5)	**68.4(-19.8)**	86.2(-2)	82.5(-5.7)	81.1(-7.1)	69.4(-18.8)	85(-3.2)	68.9(-19.3)	81.2(-7)
		Attack-4	86	24.8(-61.2)	34(-52)	24.4(-61.6)	24.1(-61.9)	34.5(-51.5)	28(-58)	24.3(-61.7)	**18(-68)**	22.1(-63.9)
	Adagrad	Attack-1	97.6	97(-0.6)	93.4(-4.2)	95.8(-1.8)	96.4(-1.2)	96.7(-0.9)	93.5(-4.1)	97(-0.6)	**73.4(-24.2)**	-
		Attack-2	97.5	89(-8.5)	93.1(-4.4)	94(-3.5)	96.1(-1.4)	96.3(-1.2)	92.1(-5.4)	95(-2.5)	**70.3(-27.2)**	-
		Attack-3	89.2	88.1(-1.1)	74.7(-14.5)	86(-3.2)	86.2(-3)	84.5(-4.7)	79(-10.2)	81.9(-7.3)	**64.2(-25)**	-
		Attack-4	82	**16(-66)**	38(-44)	34.3(47.7)	32(-50)	38.4(-43.6)	20(-62)	**16(-66)**	17.8(-64.2)	-
	RMSProp	Attack-1	99	98.7(-0.3)	97.4(-1.6)	98.6(-0.4)	92.8(-6.2)	93.5(-5.5)	97(-2)	96.6(-2.4)	**83.6(-15.4)**	-
		Attack-2	98.9	98.2(-0.7)	97(-1.9)	98.3(-0.6)	89.4(-9.5)	87.7(-11.3)	91.9(-7)	91.3(-7.6)	**78.6(-20.4)**	-
		Attack-3	93.5	90.8(-2.7)	83.4(-10.1)	92.9(-0.6)	85.3(-8.2)	91(-2.5)	91.7(-1.8)	91.05(-2.4)	**72.5(-21)**	-
		Attack-4	88	**7(-81)**	31(-57)	36(-52)	32(-56)	36.2(51.8)	30(-58)	28(-60)	22.6(-65.4)	-
FMNIST	SGD	Attack-1	82.4	**71(-11.4)**	74.6(-7.4)	76.7(-5.7)	80.5(-1.9)	77.5(-4.9)	77(-5.4)	81.9(-0.5)	74.8(-7.6)	75.1(-7.3)
		Attack-2	82.1	70.3(-11.8)	**69.8(-12.3)**	74.4(-7.7)	77.2(-4.9)	79.1(-3)	76.9(-5.2)	81.1(-1)	70.2(-11.9)	71.6(-10.5)
		Attack-3	76.8	64.1(-12.7)	**63.2(-13.6)**	69.8(-7)	71.9(-4.9)	71.1(-5.7)	69.7(-7.1)	68.3(-8.5)	65.6(-11.2)	65.8(-11)
		Attack-4	72	**9(-63)**	16(-56)	32.8(-39.2)	36.2(-35.8)	44(-28)	36(-36)	32(-40)	26.1(-45.9)	23.8(-48.2)
	Adagrad	Attack-1	83.6	80.2(-3.4)	78.2(-5.4)	81.1(-2.5)	81(-2.6)	80.6(-2)	78.9(-4.7)	82.5(-1.1)	**73.2(-10.4)**	-
		Attack-2	82.2	80(-2.2)	77.5(-4.7)	80.9(-1.3)	81(-1.2)	80.2(-2)	78.1(-4.1)	82(-0.2)	**71.2(-11)**	-
		Attack-3	75.1	73.4(-1.7)	71.3(-3.8)	71.1(-4)	69.9(-5.2)	70(-5.1)	72.1(-3)	74.2(-0.9)	**69.4(-5.7)**	-
		Attack-4	80	68(-12)	**24(-56)**	36.4(-43.6)	**24(-56)**	36.8(-43.2)	34(-46)	36(-44)	26.3(-53.7)	-
	RMSProp	Attack-1	74.2	69.9(-4.3)	72.3(-1.9)	67.8(-6.4)	66(-8.2)	64.3(-9.9)	67.5(-6.7)	73.6(-0.6)	**63.9(-10.3)**	-
		Attack-2	73.9	69.4(-4.5)	71.7(-2.2)	66.7(-7.2)	66.5(-7.4)	63.8(-10.1)	66.1(-7.8)	73(-0.9)	**60.8(-13.1)**	-
		Attack-3	68.2	61.3(-6.9)	**55.8(-12.4)**	59(-9.2)	58.3(-9.9)	59.6(-8.6)	59.3(-8.9)	64.6(-3.6)	58.1(-10.1)	-
		Attack-4	69	**12(-57)**	16(-53)	38(-31)	14(-55)	32.3(-36.7)	34(-35)	32(-37)	24.7(-44.3)	-
CIFAR-10	SGD	Attack-1	79	68.5(-10.5)	**62.2(-16.8)**	78.3(-0.7)	77.8(-1.2)	76.3(-2.7)	75.2(-3.8)	73.6(-5.4)	68.9(-10.1)	69.1(-9.9)
		Attack-2	78.6	68.2(-10.4)	**61.1(-17.5)**	78.1(-0.5)	77.4(-1.2)	74.3(-4.3)	76.7(-1.9)	74.2(-4.4)	63.1(-15.5)	67.6(-11)
		Attack-3	74.3	67.4(-6.9)	60.9(-13.4)	73.4(-0.9)	73.2(-1.1)	71.5(-2.8)	71.2(-3.1)	72.8(-1.5)	**60.4(-13.9)**	69.6(-4.7)
		Attack-4	75.6	31(-44.6)	28(-47.6)	33.9(-41.7)	32.6(-43)	30(-45.6)	29.8(-45.8)	**23.4(-52.2)**	25.8(-49.8)	30.9(-44.7)
	Adagrad	Attack-1	74.2	65(-9.2)	**61(-13.2)**	73.8(-0.4)	73.1(-1.1)	72.4(-1.8)	73(-1.2)	70(-4.2)	65.8(-8.4)	-
		Attack-2	73.7	64.3(-9.4)	**60(-13.7)**	73.1(-0.6)	72.6(-1.1)	72.2(-1.5)	72.8(-0.9)	69.5(-4.2)	63.1(-10.6)	-
		Attack-3	67.4	56.9(-10.5)	**53(-14.4)**	62.4(-5)	62.2(-5.2)	60.4(-7)	61.5(-5.9)	58.4(-9)	60.4(-7)	-
		Attack-4	70.1	17(-53.1)	**13(-57.1)**	28.4(-41.7)	25(-45.1)	27.1(-43)	25.2(-44.9)	12(-58.1)	26.2(-43.9)	-
	RMSProp	Attack-1	68.2	58.6(-9.6)	**55(-13.2)**	65.8(-2.4)	64.2(-4)	65.3(-2.9)	64(-4.2)	62.1(-6.1)	62.3(-5.9)	-
		Attack-2	67.9	57.3(-10.6)	**53.2(-14.7)**	65.1(-2.8)	63.6(-5.3)	61.9(-6)	62.4(-5.5)	60.8(-7.1)	60.6(-7.3)	-
		Attack-3	63.7	56(-7.7)	**52.9(-10.8)**	62.3(-1.4)	60.8(-2.9)	61.7(-2)	59(-4.7)	59.4(-4.3)	57.2(-6.5)	-
		Attack-4	65.6	23(-42.6)	**12(-53.6)**	34(-31.6)	32.1(-33.5)	32.7(-32.9)	30.3(-35.3)	25.6(-40)	21.9(-43.7)	-

5.2.3.2 Attacks with countermeasure

The attack recall for all the optimizers, countermeasures, and datasets considering the four attacks is shown in Table 5.2.4. It is observed that countermeasures on Attack 4 decrease the attack recall significantly. Generally, KD and MCD are considered the best mitigation with all datasets because they decrease the attack recall more than other countermeasures.

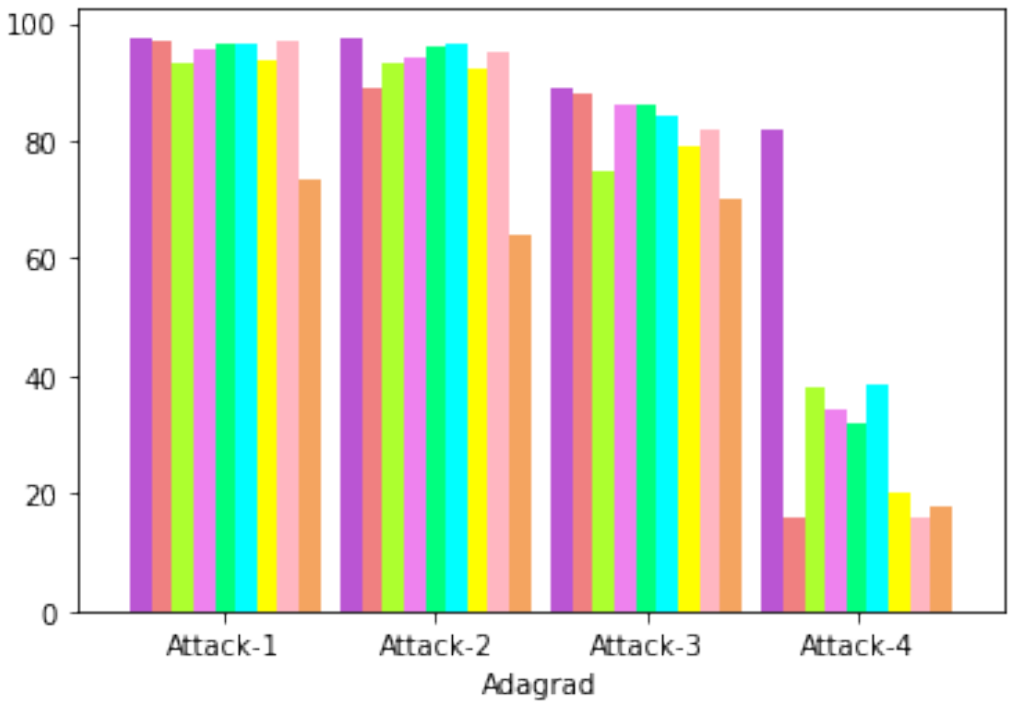


Fig. 5.2.10: A comparison of the four attacks using Adagrad optimizers with and without countermeasures - MNIST

5.2.3.3 Privacy and Utility

It is noted that KD has significant advantages over other countermeasures in handling the trade-off between privacy and utility. In general, KD decreases attack accuracy with negligible utility loss.

5.3 Conclusion

In this chapter, we studied our experimental setup with the results. We discussed the datasets, their preprocessing, the architecture of the model, and its training settings. We also discussed the comparison techniques and evaluation metrics applied to our experiments. We first compared CL and FL with and without countermeasures. In our comparison of the two approaches, CL outperformed FL regarding accuracy, which is expected. The recall in FL is similar or lower compared with the recall in CL, considering different mitigation techniques. Then we provided a comparative analysis

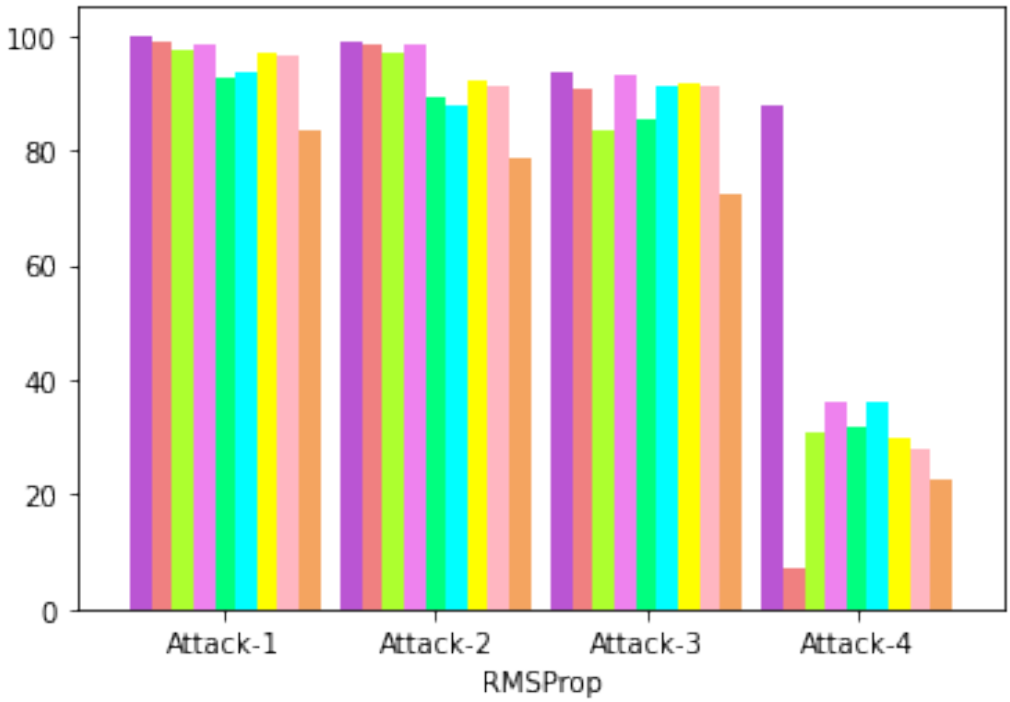


Fig. 5.2.11: A comparison of the four attacks using RMSProp optimizers with and without countermeasures - MNIST.

of all the attacks in the FL setting. The experimental results show that Attack 4 using prediction sensitivity is the worst for attackers. Among all the countermeasures, Knowledge Distillation has significant advantages in handling the trade-off between privacy and utility.

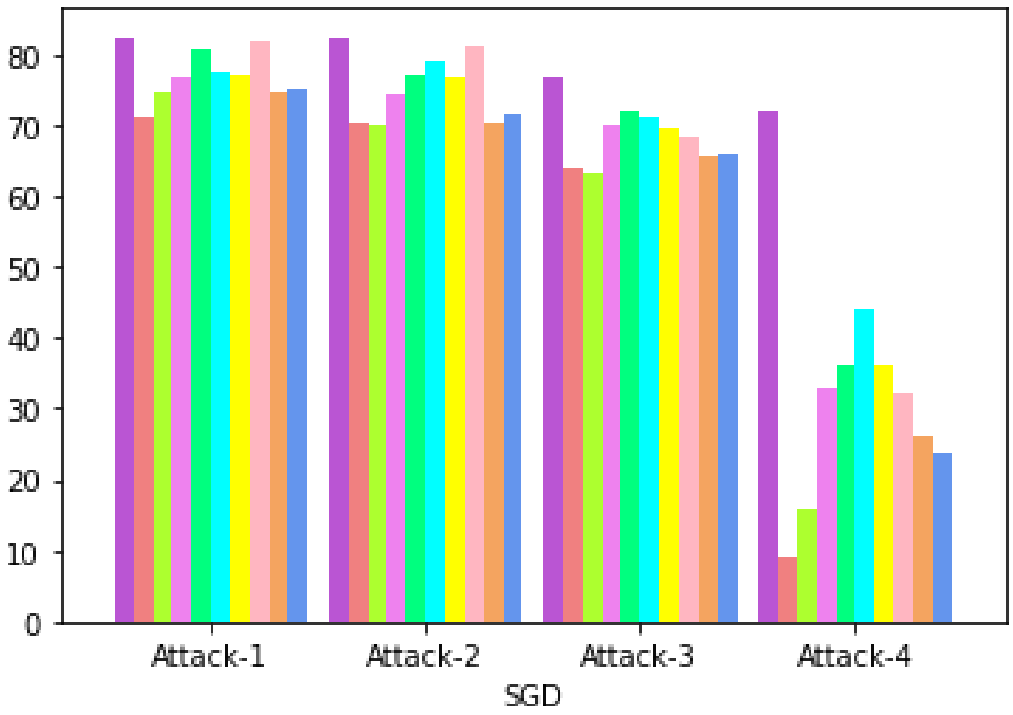


Fig. 5.2.12: A comparison of the four attacks using SGD optimizers with and without countermeasures - FMNIST

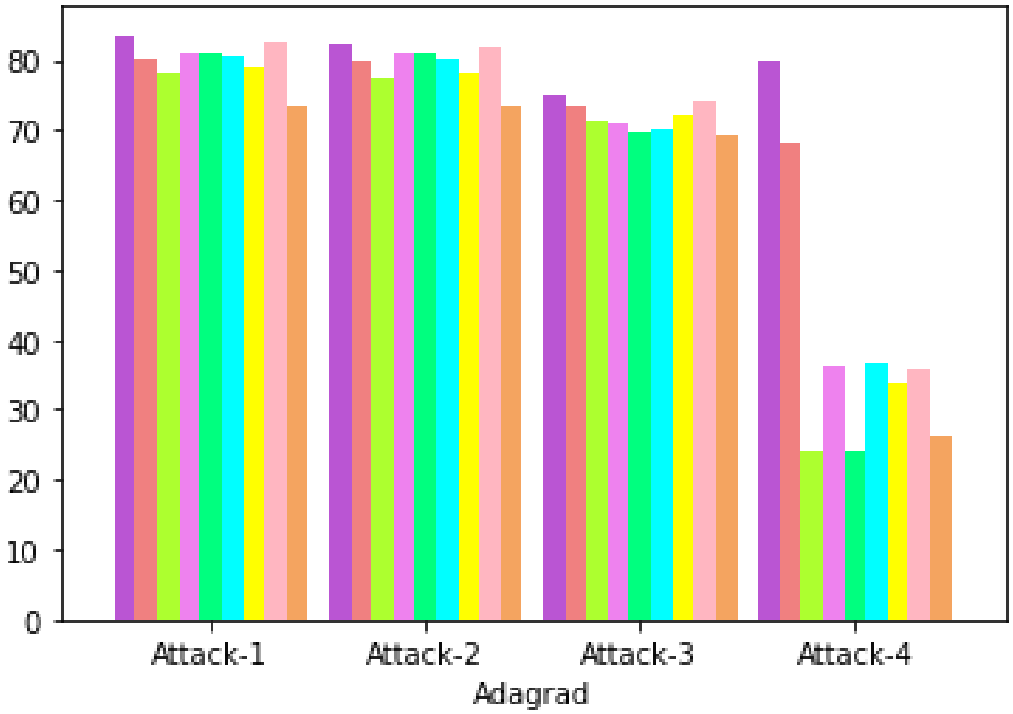


Fig. 5.2.13: A comparison of the four attacks using Adagrad optimizers with and without countermeasures - FMNIST

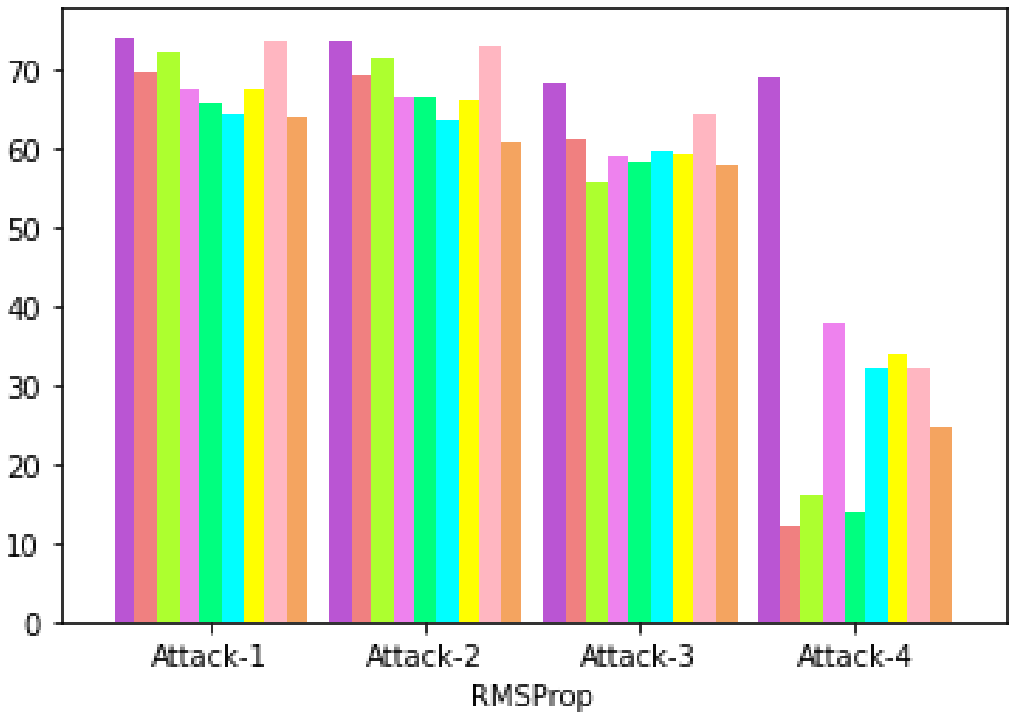


Fig. 5.2.14: A comparison of the four attacks using RMSProp optimizers with and without countermeasures - FMNIST

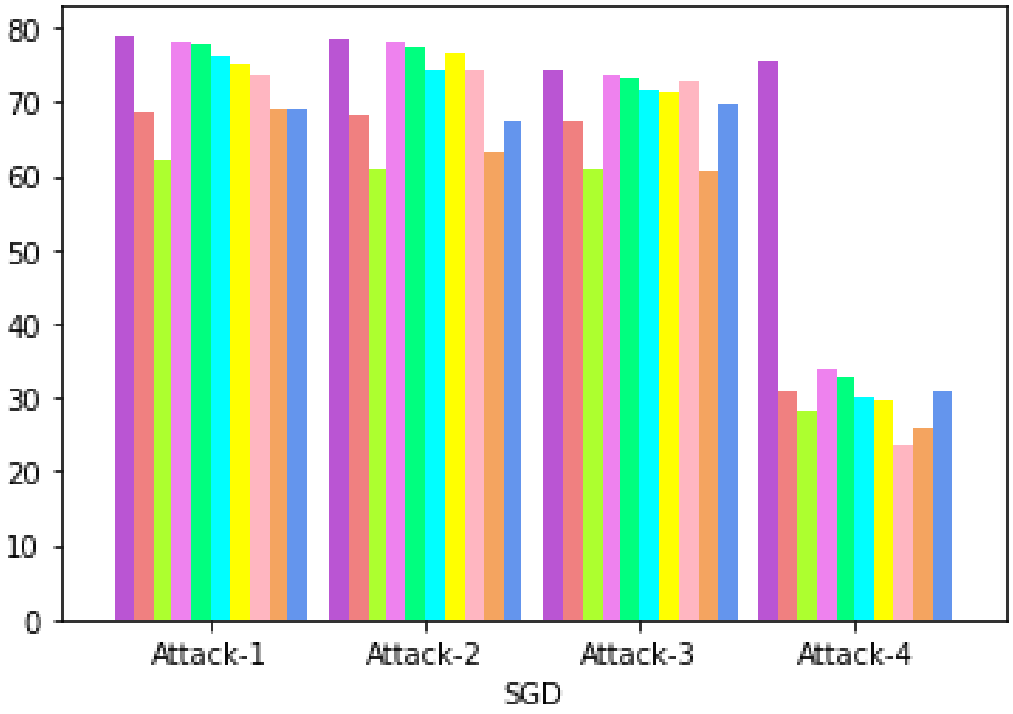


Fig. 5.2.15: A comparison of the four attacks using SGD optimizers with and without countermeasures - CIFAR-10

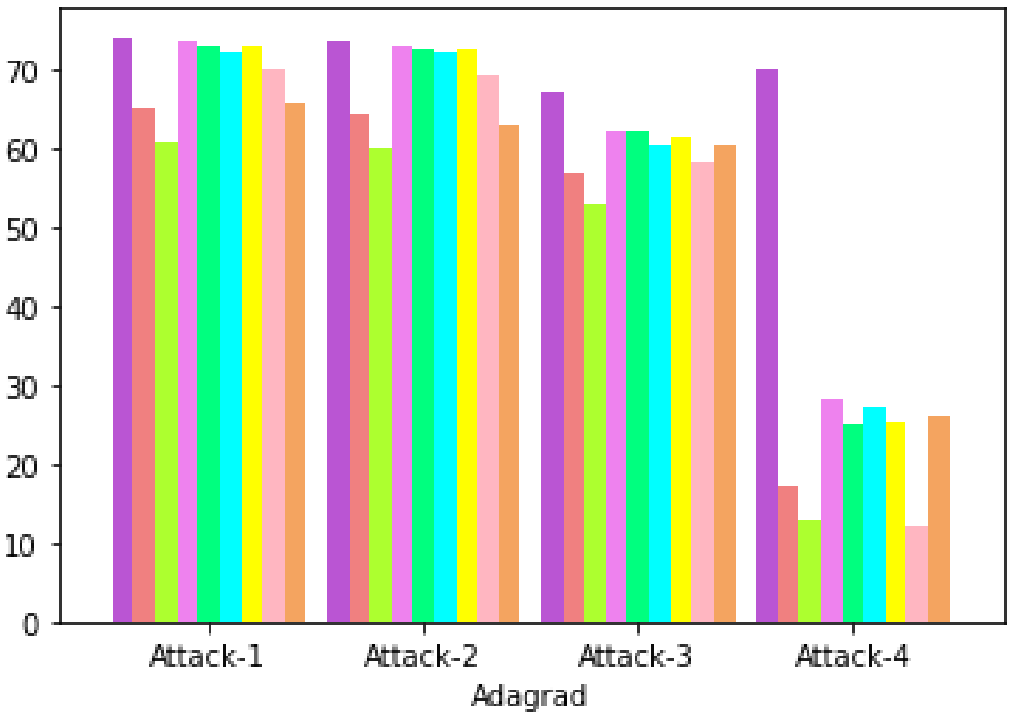


Fig. 5.2.16: A comparison of the four attacks using Adagrad optimizers with and without countermeasures - CIFAR-10

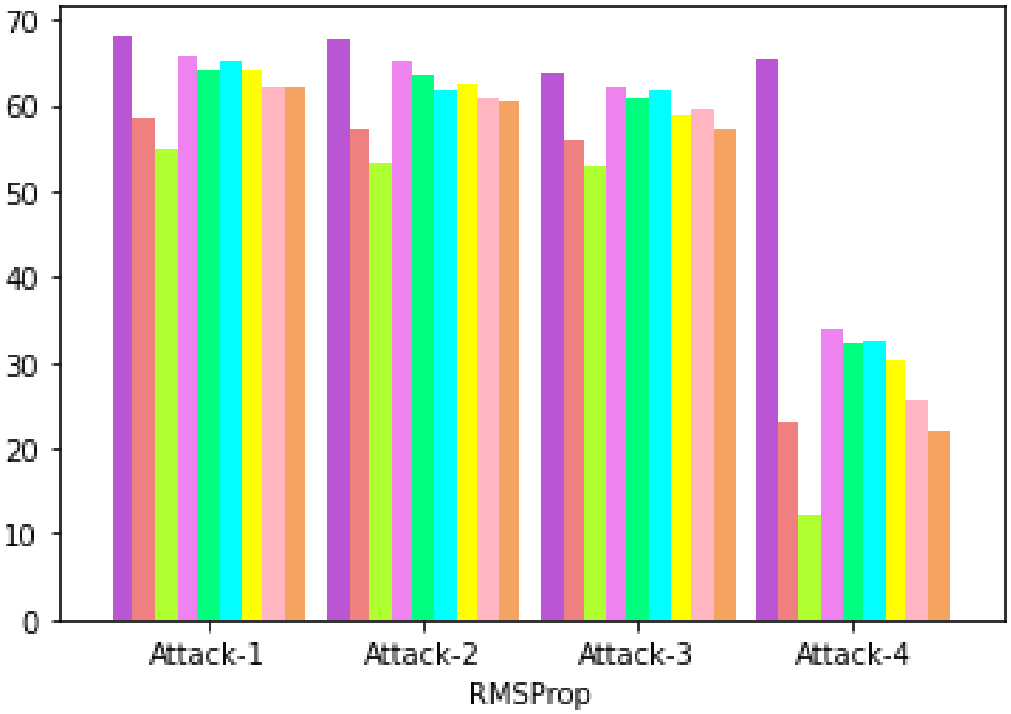


Fig. 5.2.17: A comparison of the four attacks using RMSProp optimizers with and without countermeasures - CIFAR-10

CHAPTER 6

Conclusion

Federated Learning improves privacy by allowing data owners to successfully train a model on their shared training data with the support of a central server without revealing their potentially sensitive data to either the central server or each other. While federated learning uses the enormous amount of data now accessible at edge devices, it increases data privacy by allowing data to be kept locally at the clients. Despite the advantages, federated learning is vulnerable to a membership inference attack, which attempts to find information about the training data used.

In this thesis, we first compared federated learning with centralized learning. In our comparison of the two approaches, centralized learning outperformed federated learning regarding the accuracy, which is expected. The recall of the attacks in federated learning is similar or lower compared with the recall of the attacks in centralized learning, considering different mitigation techniques. In centralized learning setting Batch Normalization and Activity Normalization does not affect the centralized model accuracy much. Knowledge Distillation and Monte-Carlo Dropout are considered the best mitigation in centralized learning settings with all datasets because they decrease the attack recall more than other countermeasures.

Next, we investigated these membership inference attacks in a federated learning scenario where the attacker tries to infer if the data is part of the training dataset and investigate these results with countermeasures. We also have provided a comparative analysis of federated learning model accuracy and the attack recall of different membership inference attacks accompanied by the effect of various countermeasures against those attacks in the federated learning environment. The experimental results

show that Attack 1 [46] is best for the attackers, whereas Attack 4 [34] is the worst for attackers. Batch Normalization, Activity Normalization, and Knowledge Distillation do not affect the federated model accuracy much. Knowledge Distillation and Monte-Carlo Dropout are the best mitigation in the federated learning environment. Among all the countermeasures, Knowledge Distillation has significant advantages in handling the trade-off between privacy and utility. This comparison is critical for future model development.

Our work uses a single countermeasure for a single experiment. As per our conclusion, future work can be done combining Knowledge Distillation with other countermeasures to analyze the effect on FL model accuracy and attack recall.

REFERENCES

[1] (1996). Health insurance portability and accountability act.

[2] (2016). General data protection regulation.

[3] (2023a). Keras documentation: Activityregularization layer.

[4] (2023b). Keras documentation: Knowledge distillation.

[5] (2023c). Keras documentation: Masking layer.

[6] (2023a). Tensorflow documentation: Activity regularization v2.11.0.

[7] (2023b). Tensorflow documentation: Batchnormalization v2.11.0.

[8] (2023c). Tensorflow documentation: Differential privacy v2.11.0.

[9] (2023d). Tensorflow documentation: Dropout v2.11.0.

[10] (2023e). Tensorflow documentation: Gaussian dropout v2.11.0.

[11] (2023f). Tensorflow documentation: Gaussian noise v2.11.0.

[12] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

[13] Alkhulaifi, A., Alsahli, F., and Ahmad, I. (2021). Knowledge distillation in deep learning and its applications. *PeerJ Computer Science*, 7:e474.

[14] Apparaju, A. and Arandjelović, O. (2022). Towards new generation, biologically plausible deep neural network learning. *Sci*, 4(4):46.

[15] Asad, M., Moustafa, A., and Ito, T. (2021). Federated learning versus classical machine learning: A convergence comparison. *arXiv preprint arXiv:2107.10976*.

[16] Backes, M., Berrang, P., Humbert, M., and Manoharan, P. (2016). Membership privacy in microrna-based studies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 319–330.

[17] Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. *Advances in neural information processing systems*, 31.

[18] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pages 177–186. Springer.

[19] Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. (2019). The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284.

[20] Conti, M., Li, J., Picek, S., and Xu, J. (2022). Label-only membership inference attack against node-level graph neural networks. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, pages 1–12.

[21] DeepAI (2019). Batch normalization.

[22] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142.

[23] Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87.

[24] Drainakis, G., Katsaros, K. V., Pantazopoulos, P., Sourlas, V., and Amditis, A. (2020). Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pages 1–8. IEEE.

[25] Dwork, C. (2011). A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95.

[26] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

[27] Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

[28] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

[29] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

[30] Jiang, D., Shan, C., and Zhang, Z. (2020). Federated learning algorithm based on knowledge distillation. In *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, pages 163–167. IEEE.

[31] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[32] Lee, H., Kim, J., Ahn, S., Hussain, R., Cho, S., and Son, J. (2021). Digestive neural networks: A novel defense strategy against inference attacks in federated learning. *computers & security*, 109:102378.

[33] Li, X., Chen, B., and Lu, W. (2023). Feddkd: Federated learning with decentralized knowledge distillation. *Applied Intelligence*, pages 1–17.

[34] Liu, L., Wang, Y., Liu, G., Peng, K., and Wang, C. (2022). Membership inference attacks against machine learning models via prediction sensitivity. *IEEE Transactions on Dependable and Secure Computing*.

[35] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

[36] McMahan, H. B. and Streeter, M. (2010). Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*.

[37] Melis, L., Song, C., De Cristofaro, E., and Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE.

[38] Nasr, M., Shokri, R., and Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE.

[39] Niknam, S., Dhillon, H. S., and Reed, J. H. (2020). Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58(6):46–51.

[40] Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*.

[41] Oleszak, M. (2021). Monte carlo dropout.

[42] Peng, S., Yang, Y., Mao, M., and Park, D.-S. (2022). Centralized machine learning versus federated averaging: A comparison using mnist dataset. *KSII Transactions on Internet and Information Systems (TIIS)*, 16(2):742–756.

[43] Poggiolini, P. (2012). The gn model of non-linear propagation in uncompensated coherent optical systems. *Journal of Lightwave Technology*, 30(24):3857–3879.

[44] Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. (2018). Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*.

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

[46] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

[47] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[48] Su, T., Wang, M., and Wang, Z. (2021). Federated regularization learning: an accurate and safe method for federated learning. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 1–4. IEEE.

[49] Tieleman, T., Hinton, G., et al. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[50] Wang, C., Deng, C., and Ivanov, V. (2020). Sag-vae: End-to-end joint inference of data representations and feature relations. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE.

[51] Wu, C., Wu, F., Lyu, L., Huang, Y., and Xie, X. (2022). Communication-efficient federated learning via knowledge distillation. *Nature communications*, 13(1):2032.

[52] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

[53] Xiao, Y., Yan, C., Lyu, S., Pei, Q., Liu, X., Zhang, N., and Dong, M. (2022). Defed: An edge feature enhanced image denoised networks against adversarial attacks for secure internet-of-things. *IEEE Internet of Things Journal*.

[54] Xie, Y., Chen, B., Zhang, J., and Wu, D. (2021). Defending against membership inference attacks in federated learning via adversarial example. In *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, pages 153–160. IEEE.

[55] Yim, J., Joo, D., Bae, J., and Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4133–4141.

[56] Zheng, J., Cao, Y., and Wang, H. (2021). Resisting membership inference attacks through knowledge distillation. *Neurocomputing*, 452:114–126.

VITA AUCTORIS

NAME:	Saroj Dayal
PLACE OF BIRTH:	Mumbai
YEAR OF BIRTH:	1992
EDUCATION:	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2023