

Gradient Leakage and Protection for Federated Learning

Hanchi Ren

845231

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Doctor of Philosophy




Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

May 11, 2023

Declaration


This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed  (candidate)

Date 14/03/2023

Statement 1


This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed  (candidate)

Date 14/03/2023

Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed  (candidate)

Date 14/03/2023

I would like to dedicate this work to my wife, Yi Hu, who gives me unwavering support and encouragement during my Ph.D studies. Thank you, my dear!

Abstract

In recent years, data privacy has become a critical issue in the field of Machine Learning (ML), given the significant amount of sensitive data involved in training and inference processes. Several approaches have been developed to address this challenge, including cryptography and collaborative training. Cryptography techniques, such as Homomorphic Encryption (HE) and Differential Privacy (DP), have gained popularity due to their ability to protect sensitive data during computation. HE allows computations to be performed directly on encrypted data without the need to decrypt it, thus ensuring privacy while still providing accurate results. On the other hand, DP adds random noise to data to protect individuals' privacy while preserving statistical accuracy. Collaborative training methods, such as Secure Multi-Party Computation (MPC), Distributed Learning, and Federated Learning (FL), aim to address privacy concerns by enabling secure local computation. In MPC, parties collaborate to compute a function without revealing their inputs to each other, making it suitable for privacy-preserving ML tasks. Distributed Learning allows data to be distributed across multiple devices or nodes, reducing the risk of data breaches while still achieving accurate results. FL enables the training of ML models on decentralised data without transferring raw data to a central location. While these techniques have proven effective in protecting sensitive data, they also have some limitations. For instance, HE and DP may be computationally expensive, which can hinder their widespread adoption. Additionally, collaborative training methods may require significant communication overhead and synchronisation, which can affect training efficiency.

Collaborative training through gradient exchange has been widely used in Deep Learning (DL) as a secure way to train a robust model. However, recent research has shown that this method may not be entirely secure. In fact, sensitive information can be recovered from the shared gradient, compromising privacy and leading to malicious actors' potential theft of valuable data. Various studies have demonstrated that the publicly shared gradient can reveal sensitive information about the training data, such as the presence of specific individuals or properties. This can lead to significant privacy breaches, especially in sensitive areas such as

healthcare or finance. As the demand for privacy-preserving ML grows, there is a need for further research and development of effective and robust techniques to ensure data privacy during collaborative training.

This thesis aims to investigate how to reconstruct private input data from the publicly shared gradient and how to prevent gradient leakage in terms of gradient-sharing protocol and a private key-lock module. We first show that in an FL system, image-based privacy data can be easily retrieved from the shared gradient through our proposed Generative Regression Neural Network (GRNN). Our attack involves formulating the problem as a regression task and optimising two branches of the generative model by minimising the gradient distance. The findings of our study demonstrate that even seemingly innocuous shared information can lead to the recovery of sensitive data. This highlights the importance of developing robust privacy-preserving techniques to protect sensitive information during collaborative ML. Our proposed GRNN attack serves as a wake-up call to the ML community to address the privacy concerns associated with FL.

Our following study found that the generalisation ability of joint models in FL is poor on Non-Independent and Non-Identically Distributed (Non-IID) data, particularly when the Federated Averaging (FedAvg) strategy is used, leading to weight divergence. To address this issue, we propose a novel boosting algorithm for FL that addresses the generalisation and gradient leakage problems, resulting in faster convergence in gradient-based optimisation. Our proposed boosting algorithm aims to improve the performance of FL models by aggregating models trained on subsets of data, addressing the weight divergence issue. The algorithm leverages an adaptive weighting strategy, where the weights of each model are adjusted based on their performance, with models that perform better receiving more weight. Additionally, we introduce a privacy-preserving component to the algorithm, where local models are encrypted to reduce the risk of gradient leakage. Our proposed boosting algorithm shows promising results in addressing FL's generalisation and gradient leakage issues, leading to faster convergence in gradient-based optimisation. The findings of our study highlight the importance of developing robust techniques to improve the performance of FL models and ensure data privacy during collaborative ML.

At last, our research proposes a new approach to defending against gradient leakage attacks in FL through a private key-lock module (FedKL). This method involves securing arbitrary model architectures with a private key-lock module, where only the locked gradient is transferred to the parameter server for aggregating the global model. The proposed FedKL method is

designed to be robust against gradient leakage attacks, ensuring that sensitive information cannot be reconstructed from the shared gradient. The key-lock module is trained in a way that, without the private information of the module, it becomes infeasible to reconstruct training data from the shared gradient. Furthermore, the inference performance of the global model is significantly undermined without the key-lock module, making it an integral part of the model architecture. Our theoretical analysis explains why the gradient can leak private information and how the proposed FedKL method defends against the attack based on our analysis. The proposed FedKL method provides a new perspective on defending against gradient leakage attacks in FL, enhancing the security and privacy of sensitive data.

We will continuously work on the privacy-preserving FL. In our previous work, we have identified a number of follow-up research point. Examples include gradient leakage for Natural Language Processing (NLP), an adaptive gradient aggregation method and partial gradient leakage. Since we have theoretically proven that the private information is carried by the gradients, so finding the state-of-the-art methods of stealing data and defending against leakage is a long-term study in safeguarding privacy.

Acknowledgements

First and foremost, I would like to express my deep gratitude to my supervisor, Prof. Xianghua Xie and Dr Jingjing Deng, for their invaluable guidance and support throughout my Ph.D studies. Their knowledge, expertise, and unwavering support have been instrumental in helping me navigate the challenges of graduate school and complete my research.

I am also grateful to Yi Hu, my wife, for her unwavering support and encouragement throughout my Ph.D journey. Her love, patience, and understanding have been my constant source of strength and motivation. Without her, I would not have been able to overcome the many obstacles and setbacks that I encountered along the way. But most of all, thank you for giving birth to our son, Ansen. Her strength and resilience during pregnancy and childbirth were truly inspiring, and I will always be grateful for the gift of our handsome child. She truly is the best partner and mother anyone could ask for, and I am so blessed to have her in my life. Thank you for everything she does for our family and for her unwavering support and love. I love her more than words could ever express.

I would also like to thank my family, my mother, father, grandmother, mother-in-law and father-in-law, for their love and support throughout my academic study career. I just wanted to take a moment to express my gratitude for their financial support for my Ph.D studies. Their generosity has allowed me to focus on my research and education without the stress and worry of financial burdens. Their belief in me and my goals has meant so much to me, and I am forever grateful for their unwavering support. I promise to make them proud by dedicating myself to my studies and using my knowledge and skills to impact the world positively.

Finally, I want to express my gratitude to the rest of my family and friends who have supported me in various ways throughout my Ph.D studies. Their encouragement, advice, and camaraderie have been invaluable in helping me stay focused and motivated. Their love and support have meant the world to me, and I am truly grateful for everything they have done for me.

Contents

List of Acronyms	ix
List of Tables	xii
List of Figures	xiv
1 Introduction	1
1.1 Motivations	1
1.2 Overview and Contributions	3
1.3 Outline	7
2 Background	9
2.1 Introduction	9
2.2 Deep Learning and Machine Learning	9
2.3 Federated Learning	13
2.4 Private Issues	15
2.5 Summary	23
3 Generative Data Leakage Attack	25
3.1 Introduction	25
3.2 Proposed Method	27
3.3 Experiment and Discussion	32
3.4 Summary	51
4 Protected Gradient Boosting	53
4.1 Introduction	53
4.2 Proposed Method	55

4.3	Experiment and Discussion	61
4.4	Summary	70
5	Leakage Defence with Key-Lock	73
5.1	Introduction	73
5.2	Proposed Method	76
5.3	Experiment and Discussion	87
5.4	Summary	98
6	Conclusions and Future Work	99
6.1	Conclusions	99
6.2	Future Work	101
6.3	The End	103
	Bibliography	105

List of Acronyms

AdaBoost	Adaptive Boosting
AI	Artificial Intelligence
AIGC	AI Generated Content
ANN	Artificial Neural Network
BiLSTM	Bidirectional Long-Short Term Memory
BN	Batch Normalisation
CD	Cosine Distance
CE	Cross Entropy
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
CTC	Connectionist Temporal Classification
DCGAN	Deep Convolutional GAN
DL	Deep Learning
DLG	Deep Leakage from Gradients
DNN	Deep Neural Network
DP	Differential Privacy
DT	Decision Tree
EEA	European Economic Area
EM	Earth Mover's

EMD	Earth Mover's Distance
EU	European Union
FC	Fully-Connected
FCNN	Fully-Connected Neural Network
FedAvg	Federated Averaging
FedBoosting	Federated Boosting
FedSGD	Federated Stochastic Gradient Descent
FL	Federated Learning
GAN	Generative Adversarial Network
GBM	Gradient Boosting Machines
GDPR	General Data Protection Regulation
GGL	Generative Gradient Leakage
GLU	Gated Linear Unit
GPU	Graphics Processing Unit
GRNN	Generative Regression Neural Network
HE	Homomorphic Encryption
HMM	Hidden Markov Model
iDLG	Improved Deep Leakage from Gradients
IG	Inverting Gradient
IID	Independent and Identically Distributed
LFW	Labelled Faces in the Wild
LPIPS	Learned Perceptual Image Patch Similarity
MAP	Maximum A Posteriori
ML	Machine Learning
MPC	Secure Multi-Party Computation
MSE	Mean Square Error
NLP	Natural Language Processing

Non-IID	Non-Independent and Non-Identically Distributed
PRECODE	Privacy Enhancing Module
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SSIM	Structural Similarity
SVM	Support Vector Machine
TEE	Trusted Execution Environment
TVLoss	Total Variation Loss
WD	Wasserstein Distance
WGAN	Wasserstein GAN
XGBoost	eXtreme Gradient Boosting

List of Tables

3.1	Construction of upsampling block.	30
3.2	Examples of data leakage attack.	33
3.3	Examples of data leakage attack.	34
3.4	Average PSNR scores achieved by DLG and GRNN.	35
3.5	Comparison of image recovery.	37
3.6	Data leakage attack with different training batch sizes.	39
3.7	Examples of failed data leakage attack.	39
3.8	Some randomly selected ground truth images and corresponding recovered images.	40
3.9	Data leakage attack using GRNN.	42
3.10	Recovered images with different resolutions using GRNN.	43
3.11	Randomly selected images recovered by GRNN and IG.	44
3.12	Comparison of image recovery.	45
3.13	Comparison of label inference accuracy (%).	46
3.14	The re-identification results.	47
3.15	Performances of different network architectures.	48
3.16	Average PSNR scores with different noise types and scales.	50
4.1	Recognition accuracies (%)	63
4.2	Visual example of testing results.	64
4.3	Recognition accuracies (%)	67
5.1	Testing accuracy(%) from models trained at different settings.	88
5.2	Testing accuracies and losses	90
5.3	Accuracy (%) of models trained by FedKL	91
5.4	Comparison of image reconstruction	93
5.5	Experiments performed on GGL.	94

5.6	Quantitative comparison	95
5.7	Comparison of image reconstruction	96

List of Figures

3.1	Illustration of how GRNN as a malicious server is deployed in a FL system.	28
3.2	Details of the proposed GRNN	29
3.3	Distances between true and generated images	36
3.4	MSE loss visualisation for DLG and GRNN with batch size 32.	37
3.5	Successful attack rate	41
3.6	MSE distances between true images and generated fake images	49
4.1	The schematic diagram of proposed FedBoosting and encryption protocol.	56
4.2	Decision boundaries	62
4.3	Visual example of training images	62
4.4	Testing accuracy of FedAvg and FedBoosting	66
4.5	Testing accuracy of FedAvg and FedBoosting	67
4.6	Performance comparison	69
5.1	Illustration of FedKL	74
5.2	Details of key-lock module	85
5.3	FL system with the key-lock module.	86

Chapter 1

Introduction

1.1 Motivations

Artificial Intelligence (AI) is an interdisciplinary field of computer science that focuses on creating machines and algorithms that can perform tasks that typically require human-like intelligence, such as Natural Language Processing (NLP), image and speech recognition, decision-making, and problem-solving. In recent years, there has been a huge advancement in the development of AI, benefited by the availability of large-scale data, powerful computing resources, and innovative algorithms. As laws improve and people become more conscious of privacy protection, it will not be appropriate to centralise data from different data providers on one side. Federated Learning (FL) is a promising approach for training Machine Learning (ML) models on distributed and decentralised datasets, without the need to centralise the data. This makes it attractive for applications in which data privacy and security are critical, such as healthcare or finance. However, FL also poses new challenges, one of which is the potential for gradient leakage. This occurs when the local model updates sent by the participating clients during the training process contain information about their private data, which could be inferred by an adversary. The typical FL approach heavily relies on a central parameter server for global model aggregation. In FL, multiple local clients collaboratively train a global model by exchanging model updates with the central server, which performs the model aggregation and updates the global model. On the other hand, FL is also a communication-intensive framework, it creates a potential attack surface for malicious. An attacker could be the parameter server. Alternatively, an attacker could hack into the communication channels and intercept the local gradients, compromising the privacy of the participants' data. Therefore, it is essential to

implement robust security measures to protect against such attacks and ensure the privacy of the participants' private data.

Gradient leakage is a significant security concern in ML, as it can allow an attacker to infer sensitive information about the data used to train a model. This can have serious consequences, such as revealing personal information about individuals or compromising the security of essential systems. Defending against gradient leakage is a challenging problem because it requires a deep understanding of both the ML model and the data it is trained on. One approach to defence is to use private training algorithms that are designed to prevent the leakage of information. These algorithms often use techniques such as Differential Privacy (DP) or Secure Multi-Party Computation (MPC) to ensure that the model can be trained without revealing sensitive information. Another approach to defence is to use robust learning algorithms resistant to attacks that try to extract sensitive information from the model. These algorithms often use adversarial training or regularisation techniques to make the model less susceptible to attacks.

Gradient leakage can undermine the privacy and security of FL; therefore, it is important to understand its mechanisms and develop effective defence strategies. To address this problem rigorously and comprehensively. The research could involve developing new theoretical frameworks for understanding and preventing gradient leakage and designing and evaluating practical defences against gradient leakage. In this thesis, I will investigate the problem of gradient leakage in FL and propose novel defence methods to prevent it. The study in this thesis will contribute to the growing body of knowledge on FL and provide practical solutions to improve its privacy and security. The impact of this research could be significant, as gradient leakage is a primary concern in the field of ML, and the development of effective defences against it could help to ensure the security of ML systems. Additionally, the research could have critical practical applications, such as developing ML systems for sensitive applications such as healthcare or finance.

The major motivations for studying gradient leakage and defence are as follows:

- To protect the privacy of individuals whose data is used to train FL models. Ensuring that sensitive information is not leaked during the training process, prevents attackers from gaining access to this information and using it to their advantage.
- To improve the security of FL models. Making models resistant to attacks that try to extract sensitive information, prevents attackers from using the model to make predictions.
- To advance the field of FL. By studying gradient leakage and defence, gain insights into

the limitations of existing algorithms and propose new techniques for training private and robust models.

1.2 Overview and Contributions

This work aims to understand FL and how gradient leakage occurs. We also aim to defend the gradient leakage to protect the private training information. In Chapter 3, we will present that, in the FL system, image-based privacy data can be easily recovered in full from the shared gradient only via our proposed Generative Regression Neural Network (GRNN). We formulate the attack as a regression problem and optimise two branches of the generative model by minimising the distance between gradients. In Chapter 4, we will show a novel boosting algorithm for FL to address both the generalisation and gradient leakage issues, as well as achieve faster convergence in gradient-based optimisation. In addition, a secure gradient-sharing protocol using Homomorphic Encryption (HE) and Differential Privacy (DP) is introduced to defend against gradient leakage attack and avoid pairwise encryption that is not scalable. In Chapter 5, we will explore a new perspective of method on defending FL gradient leakage by securing arbitrary model architectures with a private key-lock module (FedKL). Only the locked gradient is transferred to the parameter server for aggregating the global model. The proposed FedKL is robust against gradient leakage attacks. In the rest of this section, we will briefly describe the work presented in each chapter. We will also discuss the connection to the motivations we discussed in Section 1.1 and the contribution of each proposed approach.

1.2.1 Gradient Leakage Attack

Generative Adversarial Network (GAN) is a generative model proposed by Goodfellow *et al.* [1] for image generation. Inspired by the GAN and Deep Leakage from Gradients (DLG) model, we introduce a gradient-guided image generation strategy that properly addresses the stability and data quality issues of DLG based methods. The proposed GRNN, a novel data leakage attack method, is capable of recovering private training images up to a resolution of 256*256 and a batch size of 256. The method is particularly suitable for FL as the local gradient, and global model are readily available in the system setting. GRNN consists of two branches for generating fake training data and corresponding labels. It is trained in an end-to-end fashion by approximating the fake gradient that is calculated by the generated data and label to the true gradient given the global model. Mean Square Error (MSE), Wasserstein Distance (WD)

and Total Variation Loss (TVLoss) are used jointly to evaluate the divergence between true and fake gradients. We empirically evaluate the performance of our method on several image classification tasks and comprehensively compare against the state-of-the-art. The experimental results confirm that the proposed method is much more stable and capable of producing better-quality images when a large batch size and resolution are used. The contributions of this work are four-fold:

- We propose a novel method of data leakage attack for FL, which is capable of recovering private training images up to a resolution of 256×256 , a batch size of 256 as well as the corresponding labels from the shared gradient. The implementation of the method is publicly available to ensure its reproducibility.
- We conduct a comprehensive evaluation, where both qualitative and quantitative results are presented to prove the effectiveness of GRNN. We also compare the proposed method against the state-of-the-art, which shows that GRNN is superior in terms of the success rate of attack, the fidelity of recovered data and the accuracy of label inference. In addition, our method is much more stable than others with respect to the size of the training batch and input resolution.
- We conduct a face re-identification experiment that shows using the image generated by the proposed GRNN can achieve higher *Top-1*, *Top-3* and *Top-5* accuracies compared to the state-of-the-art.
- We discuss the potential defence strategies and quantitatively evaluate the effectiveness of our method against noise addition defence strategy.

The findings of this chapter, including methodology and experimental results, are presented in the following publication:

- Ren, H., Deng, J., & Xie, X. (2022). GRNN: generative regression neural network — a data leakage attack for federated learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4), 1-24.

1.2.2 Gradient Protected Boosting

In Chapter 4, we propose the Federated Boosting (FedBoosting) method to address the weight divergence and gradient leakage issues in the general FL framework. Instead of treating

individual local models equally when the global model is aggregated, we consider the data diversity of local clients in terms of the status of convergence and the ability of generalisation. Hence, the different client is given a different aggregating percentage instead of averaging the gradients from those clients. To address the potential risk of data leakage via shared gradients, a DP based linear aggregation method is proposed using Homomorphic Encryption (HE) [2] to encrypt the gradients, which provides two layers of protection. The proposed encryption scheme only leads to a negligible increase in computational cost.

The proposed method is evaluated using a text recognition task on public benchmarks, as well as a binary classification task on two datasets, which demonstrates its superiority in terms of convergence speed, prediction accuracy and security. The performance reduction due to encryption is also evaluated. Our contributions are three-fold:

- We propose a novel aggregation strategy, namely FedBoosting for FL to address the weight divergence and gradient leakage issues. We empirically demonstrate that FedBoosting converges significantly faster than Federated Averaging (FedAvg) while the communication cost is identical to traditional approaches. Especially when the local models are trained with a small batch size and the global model is aggregated after a large number of epochs, our approach can still converge to a reasonable optimum, whereas FedAvg often fails in such a case. Our implementation is publicly available to ensure reproducibility. It can also be run in a distributed multiple Graphics Processing Units (GPUs) setup. ¹
- We introduce a dual layer protection scheme using HE and DP to encrypt gradients flowing between server and clients, which protect the data privacy from gradient leakage attack.
- We show the feasibility of our method on two datasets by evaluating the decision boundaries visually. Furthermore, we also demonstrate its superior performance in a visual text recognition task on multiple large-scale Non-Independent and Non-Identically Distributed (Non-IID) datasets compared to the centralised approach and FedAvg. The experimental results confirm that our approach outperforms FedAvg in terms of convergence speed and prediction accuracy. It suggests that FedBoosting strategy can be integrated with other Deep Learning (DL) models in the privacy-preserving scenarios.

¹<https://github.com/Rand2AI/FedBoosting>

The findings of this chapter, including methodology and experimental results, are presented in the following publication:

- Ren, H., Deng, J., Xie, X., Ma, X., & Wang, Y. (2022) FedBoosting: Federated Learning with Gradient Protected Boosting for Text Recognition. IEEE Transactions on Emerging Topics in Computing (TETC). Under Reviewing.

1.2.3 Gradient Leakage Defence with Key-Lock Module

In Chapter 5, we first theoretically approve that the feature maps computed from the fully-connected layer, convolutional layer and Batch Normalisation (BN) layer contain the private information of input data, where such information also co-exists in the gradient at the backward passing stage. Furthermore, we hypothesise that the gradient leakage attack is possible only when the gradient spaces between the global and local models are well aligned. Therefore, we propose *FedKL*, a key-lock module which is capable of differentiating, misaligning and locking the gradient spaces with a private key meanwhile keeping the federated aggregation the same as the typical FL framework. In summary, we reformulate the scale and shift processes in the normalisation layer. A private key, *i.e.*, randomly generated sequence, is fed into two fully-connected layers, and the outputs are the privately owned coefficients for the scale and shift processes. Both theoretical analysis and experimental results show that the proposed key-lock module is feasible and effective in defending against the gradient leakage attack as the consistency of private information in the gradient is obfuscated so that the malicious attacker cannot formula the forward-backwards propagation without the private key and the gradient of the lock layer. Therefore, it is no longer feasible to reconstruct local training data by approximating the shared gradient in the FL system. Our theoretical and experimental results suggest that the *FedKL* brings the benefits in four aspects:

- Safe - a strong protection strategy against the gradient leakage attack;
- Accurate - negligible degradation of inference performance compared to the model without the key-lock module;
- Efficient - additional computational cost on the key-lock module is negligible;
- Flexible - applicability to arbitrary network architecture.

The findings of this chapter, including methodology and experimental results, are presented in the following publication:

- Ren, H., Deng, J., Xie, X., Ma, X., & Ma, J. (2022) FedKL: Gradient Leakage Defence with Key-Lock Module for Federated Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). In Preparation.

1.3 Outline

The rest of the thesis is organised as follows:

- In Chapter 2, background knowledge is presented, providing the necessary information that includes the introduction to FL, gradient leakage and gradient leakage defence. Some classic and state-of-the-art methods are also reported in this chapter.
- In Chapter 3, a state-of-the-art gradient leakage method, GRNN, is introduced. The proposed method utilised two branches of the generative GAN-based model to generate fake images and labels. Then regress the fake gradient to the true gradient for optimising the model to enhance its generative ability.
- In Chapter 4, a novel boosting algorithm is introduced for FL to address both the generalisation and gradient leakage issues, as well as achieve faster convergence in gradient-based optimisation. To achieve the gradient leakage defence, HE and DP are utilised to secure the gradient transmitting between the client and server.
- In Chapter 5, a new perspective of method on defending Federated Learning Gradient Leakage by securing arbitrary model architectures with a private key-lock module (FedKL) is presented. A key sequence is used as the input of two lock layers. Then the lock layers will output two parameters, one for scale and the other for the shifting process. Hence, the inheritance of the private input information throughout the gradient is broken. On the other hand, we theoretically analyse how the gradient leakage takes place in the fully-connected layer, convolutional layer and BN layer. Mathematical reasoning processing is provided.
- In Chapter 6, a comprehensive discussion on the proposed approaches and possible extensions of the work presented in this thesis is reported.

Chapter 2

Background

2.1 Introduction

In this chapter, we will look through this thesis’s background knowledge. In Section 2.2, a comprehensive look back on DL, ML, Convolutional Neural Network (CNN) and image classification are reported. In Section 2.3, we first introduce the basic concept of FL, FedAvg, which is the core technique in the thesis. Then, an overview of gradient leakage in FL is introduced, which includes many state-of-the-art attack methods. At last, we investigate how to defend against gradient leakage attack, such as gradient perturbation, data obfuscation or sanitization, DP, HE and MPC. Three state-of-the-art defence methods are reported.

2.2 Deep Learning and Machine Learning

DL and ML are two closely related fields within the broader domain of AI. Both are concerned with developing algorithms and systems that can learn and adapt to new data. Both have seen tremendous advances in recent years due to the availability of large amounts of data and powerful computing resources. DL, in particular, has garnered a lot of attention in recent years due to its ability to achieve state-of-the-art results in a wide range of tasks, including image and speech recognition [3–7], NLP [8–12], and machine translation [13–17]. This success is largely due to the use of Deep Neural Network (DNN)s, which are inspired by the structure and function of the human brain and can learn to recognise patterns in data by adjusting the weights of their many interconnected nodes. DL is a more general term that encompasses a wide range of techniques for learning from data. These techniques range from simple linear regression

models to more complex algorithms such as Support Vector Machine (SVM) and Decision Tree (DT). DL algorithms can be divided into two main categories: supervised learning, in which the algorithm is trained on labelled data and makes predictions based on that training, and unsupervised learning, in which the algorithm is given unlabelled data and must find patterns and relationships within the data on its own. Unlike DL, which relies on large amounts of data and complex models to learn, ML algorithms typically require explicit feature engineering and manual labelling of the data. This means that the data needs to be structured and processed in a specific way for the algorithm to learn from it.

Some of the key applications of ML include predictive modelling [18–20], fraud detection [21–23], and recommendation [24–26]. For example, ML algorithms have been used to predict stock prices, identify fraudulent transactions, and recommend products or content to users. While ML has proven effective in many tasks, it is not as flexible or powerful as DL in some areas. ML algorithms are generally less effective at learning complex patterns and relationships in unstructured data and may require more data and computational resources to achieve similar results. Both DL and ML have a wide range of applications in various industries, including healthcare [27–29], finance [30–32], and transportation [33–35]. In healthcare, for example, DL algorithms have been used to analyse medical images and predict the likelihood of certain diseases. In contrast, ML algorithms have been used to analyse patient data and predict the likelihood of certain outcomes. In finance, ML algorithms have been used to identify fraudulent transactions and predict market trends, while in transportation, ML algorithms have been used to improve the efficiency of self-driving cars.

DL and ML are two exciting and rapidly evolving fields within AI that have the potential to revolutionise a wide range of industries and applications. While both require a strong foundation in mathematics and computer science, they offer many tools and frameworks that make it easier for developers to build and deploy ML models. As data and computing resources become more widely available, we expect to see even more advances in these fields in the coming years.

2.2.1 Convolutional Neural Network

CNNs are a type of artificial neural network specifically designed for image and video recognition tasks [36–39]. They are inspired by how the visual cortex processes information in the human brain. They have proven to be very effective in a wide range of applications, including image classification [40–44], object detection [45–49], and video analysis [50–53].

At a high level, a CNN consists of an input layer, one or more hidden layers, and an output

layer. The hidden layers consist of convolutional layers, which apply a set of learned filters to the input data, and pooling layers, which down-sample the data to reduce its dimensionality and increase the model's generalisation ability. The output layer produces a prediction based on the processed input data. One key feature of CNNs is convolutional layers, which apply filters to the input data to extract meaningful features. These filters are learned during the training process and can detect edges, lines, and other patterns in the data. The filters are applied to the input data using convolution, which involves sliding the filter over the input data and performing element-wise multiplication followed by a sum. This process results from a feature map, which represents the presence of the learned features in the input data. Another critical feature of CNNs is the use of pooling layers, which down-sample the data to reduce its dimensionality and increase the model's ability to generalise. There are several types of pooling, including max pooling and average pooling, but max pooling is the most commonly used. Max pooling works by dividing the input data into a set of non-overlapping windows and selecting the maximum value in each window. This process reduces the size of the data, while still preserving the most essential features. In addition to convolutional and pooling layers, CNNs can also include fully-connected layers, similar to the hidden layers in a traditional neural network. These layers take the output of the convolutional and pooling layers and apply a set of weights to produce a prediction. The output layer is typically a softmax layer, which produces a probability distribution over the possible classes. Training a CNN involves feeding the model a large dataset of labelled examples and adjusting the weights and biases of the layers to minimise the error between the predicted and true labels. This is done using an optimisation algorithm, such as stochastic gradient descent or Adam [54], and requires a large amount of computing power and data.

CNNs have proven to be very effective for image and video recognition tasks and have achieved state-of-the-art results on a wide range of benchmarks. They are widely used in industry and research and are an important tool in the field of DL.

2.2.2 Generative Adversarial Network

GAN is a generative model proposed by Goodfellow *et al.* [1] for image generation. It is a class of machine learning framework that has gained significant attention in recent years for its ability to generate realistic data samples. GANs consist of two neural networks, a generator and a discriminator, that are trained simultaneously in a game-theoretic setting. The generator creates synthetic data samples, while the discriminator evaluates the authenticity of these samples

against real data [55]. One of the key inspirations for GANs comes from game theory, where two players engage in a zero-sum game [56]. In the context of GAN, the generator aims to produce realistic samples to fool the discriminator, while the discriminator tries to accurately distinguish between real and fake data [57]. This adversarial process allows the generator to improve its output iteratively, ultimately creating highly realistic synthetic data. Since their inception, GANs have been used in a wide array of applications, including image synthesis [58–61], video generation [62–65], style transfer [66–68], data augmentation [69–72], and even drug discovery [73, 74]. Some notable variants and advancements of GANs include Deep Convolutional GAN (DCGAN) [75], Wasserstein GAN (WGAN) [76], and CycleGAN [77], each of which address specific challenges or extend the capabilities of the original GAN architecture. WGAN introduces the Wasserstein distance as an alternative loss function, improving training stability and addressing the mode collapse issue. CycleGAN enables unpaired image-to-image translation by enforcing cycle consistency, allowing for transformations between different data domains without paired training samples. Other techniques, such as unrolled GAN [78], focus on improving the optimisation process by unrolling the discriminator’s optimisation steps during generator updates, which helps stabilise training. In addition, studies on GAN convergence [79] have provided valuable insights into the choice of training methods and hyperparameters for more reliable GAN training. Amortised Maximum A Posteriori (MAP) inference [80] is another technique that incorporates GAN into Bayesian inference for tasks like image super-resolution, demonstrating the versatility of GAN in various machine learning applications.

GANs are a powerful class of generative models that leverage adversarial training to create realistic synthetic data. They have been applied to numerous domains and have seen significant advancements since their introduction in 2014. However, challenges remain, and ongoing research seeks to further improve their stability and performance.

2.2.3 Image Classification

Image classification assigns a class label to an input image [81]. It is a fundamental problem in computer vision and has numerous practical applications, such as in object recognition [36–38, 82], facial recognition [83–85], and autonomous vehicles [86–88]. The goal of image classification is to learn a function that maps an input image to a class label. The input to the function is an image, and the output is a class label. For example, given an image of a cat, the function should output the class label "cat". There are several approaches to image classification, including traditional ML methods and DL methods. Traditional ML

methods for image classification involve extracting features from the image and then training a classifier on these features. These features could be pixel values, edges, textures, or other image characteristics. The classifier could be a SVM, a DT, or another type of classifier. DL methods, on the other hand, do not require explicit feature extraction. Instead, they learn features directly from the raw image data using neural networks. These methods have been shown to be very effective for image classification, particularly when using CNNs. A large dataset of labelled images is required to train a DL model for image classification, such as ImageNet [89]. The model is trained by presenting it with an image and the corresponding class label and adjusting the model's parameters so that it correctly predicts the class label for that image. This process is repeated for many images in the training dataset until the model performs well. Once the model is trained, it can classify new images. Given an input image, the model predicts the class label that it belongs to. There are many challenges in image classification, including the variability of images, the need for large amounts of labelled data, and the difficulty of obtaining high-quality labels. However, with the advent of DL and the availability of large datasets and computational resources, significant progress has been made in the field, and image classification has become a critical component of many practical applications.

2.3 Federated Learning

FL is a distributed ML approach that enables the training of a ML model on multiple decentralised devices, such as smartphones or edge computing devices, without the need to share the data with a central server [90–94]. This approach aims to preserve the privacy of the data and avoid the risks associated with data leakage, while still allowing the model to learn from a large and diverse dataset. FL can be seen as an extension of traditional ML techniques, which rely on the centralisation of data in a single location for training. However, in many cases, it is not possible or desirable to centralise the data due to privacy concerns or the difficulty of collecting and storing large amounts of data from multiple sources. FL addresses this issue by allowing the training to be performed on the devices themselves without the need to transmit the data to a central server. The main idea behind FL is to train a model on multiple devices using their local data and then aggregate the model updates from each device to improve the overall model performance. This is done through a process known as FedAvg [93], where the model updates from each device are averaged together to create a global model. In FedAvg, the global model takes the average of gradients from local models, *i.e.* $\omega' = \sum_i^N \frac{1}{N} \omega_i$, where ω' and ω_i are the gradients of global model and the i_{th} local model, and N is the total number of clients.

The method is evaluated on the MNIST benchmark and demonstrates its feasibility on the classic image classification task using CNNs as the learning model. Although the experimental results show that FedAvg is suitable for both Independent and Identically Distributed (IID) and Non-IID data, it is still a statistical challenge of FL when a local model is trained on large-scale Non-IID data. In Chapter 4.3, our experimental results also support such an argument, where the prediction accuracy and convergence rate drop significantly with large-scale Non-IID data on FedAvg. FL has been applied to a variety of tasks, including NLP, image classification, and recommendation systems. It has also been used in healthcare applications, such as predicting patient outcomes or identifying early signs of disease. One of the key challenges in FL is the limited availability of data on each device, which can lead to poor model performance. To address this issue, researchers have proposed various approaches, such as transfer learning [95–97] and data augmentation [98–100], to improve the performance of the model. Another challenge is the variability in the data distribution across devices, which can lead to uneven model performance. To address this issue, researchers have proposed methods such as federated transfer learning [101–103], where a pre-trained model is fine-tuned on each device using its local data, and federated meta-learning [104–106], where the model is trained to adapt to different data distributions. FL has the potential to revolutionise how ML is performed by enabling the training of models on decentralised data without the need for data centralisation. It can potentially enable the development of personalised and privacy-preserving ML applications and democratise access to ML by allowing individuals and organisations to contribute their data to the training process. However, FL also brings new challenges and research directions, such as optimising communication and computation resources, developing robust and efficient aggregation algorithms, and designing privacy-preserving protocols. Overall, FL represents a promising approach to enable the training of ML models on decentralised data, while preserving the privacy of the data and avoiding the risks associated with data leakage. It can potentially enable the development of personalised and privacy-preserving ML applications and democratise access to ML by allowing individuals and organisations to contribute their data to the training process.

2.3.1 Horizontal, Vertical and Transfer Federated Learning

There are three main types of FL: horizontal FL, vertical FL, and transfer FL. Horizontal FL involves training a ML model using data from a federation of distributed devices or silos, where each device or silo holds a distinct and non-overlapping subset of the data. For example, a

hospital network may have data from multiple hospitals, each with its own electronic medical record system. In horizontal FL, a model is trained using data from all the hospitals without centralising the data. This can be useful in situations where data is distributed across organisations, and there are privacy or regulatory concerns around centralising the data.

Vertical FL is a technique for training ML models on decentralised data, where each participating node has a different feature space. An example of this might be training a model to predict patient outcomes, where each node has patient data from a different hospital, and each hospital has collected different types of data (e.g. one hospital has collected genetic data, while another has collected imaging data). In vertical FL, the goal is to train a model that can use the participating nodes' diverse feature spaces, without exchanging raw data between the nodes. Vertical FL has the potential to improve the accuracy and performance of machine learning models. By training models using data from multiple sources, organisations can gain access to a wider range of data and more diverse training examples. This can lead to more robust and accurate models that can better generalise to new data.

Transfer FL offers a solution to this problem by allowing multiple parties to contribute their data to the training process without actually sharing the data itself. This is achieved through the use of a central server, which coordinates the training process and manages the model updates. Each party, or client, sends their data to the server, which uses it to update the model. The updated model is then sent back to the client, who can use it to make predictions on their own data. There are several benefits to using transfer FL. First, it allows multiple parties to contribute to the training process without compromising their data privacy. This can lead to more accurate and robust models, as they can access a wider range of data. Additionally, transfer FL can reduce the need for data storage and processing resources, as the data remains on the client's systems and is only used to update the model. One of the critical challenges in implementing transfer FL is ensuring that the model updates are consistent across all clients. Achieving this can be difficult, as each client may have different data distributions and patterns. To address this issue, transfer FL relies on techniques such as federated averaging and federated optimisation to ensure that the model updates are consistent and accurately reflect the combined data from all clients.

2.4 Private Issues

In the context of FL, private issues refer to concerns related to the privacy of the data used in the training process. Specifically, one of the major challenges in FL is the risk of gradient

leakage, which occurs when the gradients transmitted between the parameter server and clients contain sensitive information about the local data of the participants. Gradient leakage refers to the unintentional exposure of a model’s training data during the model’s training process. When training on sensitive data, gradient leakage can result in privacy violations and the disclosure of private information. Several privacy-preserving techniques, including DP, MPC, and HE, have been proposed to address this issue. However, these techniques can introduce additional computational and communication overhead, which can hinder the performance of FL. Therefore, finding a balance between privacy and efficiency remains a significant challenge in the field of FL, particularly when dealing with sensitive data.

2.4.1 Secure Multi-Party Computation

FL for privacy-preserving ML is proposed for training a model across multiple decentralised edge devices or clients holding local data samples. More specifically, the FL framework keeps the raw data to the owners and trains the model locally at client nodes individually. In contrast, the gradients of those models are exchanged and aggregated instead of data. Other than FL, MPC [107, 108] is another distributed training method that ensures a high level of security at the price of expensive cryptography operations. MPC is a sub-field of cryptography that deals with the design and implementation of protocols that allow multiple parties to jointly compute a function on their private inputs without revealing those inputs to each other. MPC protocols can be used to solve many problems, including secure computation of arbitrary functions, secure data sharing, voting, auctions, and more. One of the main motivations for MPC is to enable secure collaboration between parties who may not fully trust each other. For example, consider a group of companies that want to jointly compute the average price of a product across all their stores without revealing the prices at individual stores. MPC protocols can be used to compute the average securely, without any party learning the prices at other stores [107]. MPC protocols can be broadly classified into two categories: interactive protocols [109, 110] and non-interactive protocols [111, 112]. Interactive protocols require the parties to communicate with each other in order to compute the function. Non-interactive protocols do not require any communication between the parties and can be implemented using only publicly available information. MPC is an active research topic. *FALCON* [113] is a 3-party protocol for efficient private training and inference of large ML models, offering high expressiveness, batch normalisation support, security guarantees, and new theoretical insights for better efficiency. Compared to existing solutions, *FALCON* is significantly faster and more communication efficient. In order to promote

the adoption of MPC in the realm of ML, *CRYPTEN* [114] is proposed, a software framework that offers widely-used MPC primitives through familiar abstractions found in contemporary machine learning frameworks. These include tensor operations, automatic differentiation, and modular neural networks. Yong *et al.* [115] proposed a new privacy-preserving FL framework called *chain-PPFL*, which uses a chained MPC technique. This framework employs two mechanisms: 1) single-masking to protect information exchange between participants, and 2) chained-communication to transfer masked information in a serial chain. They tested the framework using two public datasets and compared training accuracy and leak defence with other methods.

MPC is a powerful tool for enabling secure collaboration between parties who may not fully trust each other. There are challenges and limitations to MPC, for example efficiency, security and performance trade-offs, complexity. MPC protocols typically introduce significant computational and communication overhead compared to non-secure methods, which can make them unsuitable for large-scale, real-time, or resource-constrained applications. Achieving stronger security guarantees often comes at the cost of reduced performance, forcing practitioners to balance between the two based on their specific use case. It is generally more complex than traditional cryptographic techniques, which can lead to increased development time and potential security vulnerabilities. In conclusion, while MPC provides a powerful framework for privacy-preserving computation, it comes with several limitations that need to be considered in the design and implementation of MPC protocols for various applications.

2.4.2 Homomorphic Encryption and Differential Privacy

Since there is no explicit data exchange, FL does not require adding noises to the data as DP [116–120], nor encrypting data into homomorphic phase to fit a homomorphic operation as HE [121–124]. DP is a framework for analysing the privacy guarantees of algorithms that process data with sensitive information. It provides a way to quantitatively measure the amount of privacy that is preserved in the data after the algorithm has been applied. The main idea behind DP is to add randomness to the data in a way that preserves the privacy of individual records while still allowing the data to be useful for statistical analysis. This is done by introducing "noise" into the data, which makes it difficult for an attacker to determine the value of any individual record with high confidence. There are several key components to DP:

- The privacy budget: This is a measure of the amount of noise that is added to the data. The larger the privacy budget, the more noise is added and the greater the protection of

2. Background

individual privacy.

- The sensitivity of the query: Different queries have different levels of sensitivity, which determines the amount of noise that needs to be added. For example, a query that asks for the average income of a group of individuals is less sensitive than a query that asks for the income of a specific individual.
- The composition property: This property states that the privacy guarantees of multiple differentially private algorithms can be "composed" to create an overall privacy guarantee for a larger system.

There are several techniques for adding noise to data in order to achieve DP. One common approach is the use of the Laplace mechanism [125], which adds noise drawn from a Laplace distribution to the output of a query. Another approach is using the Exponential mechanism [126], which adds noise drawn from an exponential distribution to the output of a query.

HE is a form of encryption that allows mathematical operations to be performed on encrypted data, without the need to decrypt the data first [121, 127]. This property enables a number of exciting and valuable applications, such as MPC, secure outsourcing of computation, and privacy-preserving data analysis. At a high level, HE works by encrypting data so that specific mathematical operations can be applied to the encrypted data. The result of these operations will be the same as if they were applied to the original, unencrypted data. For example, if having two numbers, 3 and 4, and we want to add them together, the result would be 7. If we instead encrypt these numbers using a HE scheme and then apply the addition operation to the encrypted numbers, the result will also be 7, even though the actual values of the numbers are hidden from view. There are two main types of HE: partially HE and fully HE. Partially HE [127] schemes only allow a single type of mathematical operation to be performed on the encrypted data, such as addition or multiplication. These schemes are relatively simple to construct, but their limited functionality makes them less useful in practice. Fully HE [128] schemes, on the other hand, allow arbitrary mathematical operations to be performed on the encrypted data. These schemes are significantly more complex and computationally intensive, but they have a much wider range of possible applications. One of the critical challenges in designing a HE scheme is finding a way to encrypt the data in such a way that the mathematical operations can be applied directly to the encrypted data, without the need to decrypt it first. This requires the use of advanced mathematical techniques, such as lattices and modular arithmetic. Despite its many potential uses, HE is still an active area of research, and many challenges remain in terms

of efficiency, security, and usability. As such, HE is not yet widely used in practice, but it is an exciting area of research that has the potential to revolutionise the way we think about data privacy and security.

2.4.3 Gradient Leakage

FL is designed for privacy-protected training as the data is kept and processed locally. However, it has been highlighted in multiple studies, *e.g.* [129–131], FL suffers from the so-called gradient leakage problem that is the private training data can be recovered from the publicly shared gradients with significantly high success rate. The gradient leakage attack is a way of revealing unauthorised private training data based on the leaked gradient. This is one of the most critical privacy-preserving threats in centralised and collaborative DL systems. One of the common ways of exposing private training data in a centralised DL system is membership inference [132–136]. Gradient leakage refers to the phenomenon in which the gradients used to update the weights of a neural network model are influenced by input data from outside the current training batch or input sequence. This can occur in various ways, such as through the use of shared memory or other types of shared resources between multiple parallel or sequential processing units. One common scenario in which gradient leakage can occur is when using mini-batch gradient descent for training a neural network in FL. In this case, the model’s gradients are updated based on a small subset of the training data (the mini-batch) at each training iteration. If the mini-batch size is small enough, the attacker can reconstruct the current mini-batch, leading to gradient leakage.

Given a trained victim model and a target label, Hitaj *et al.* [129] proposed a GAN-based data recovery method that can generate a set of new data having close distribution to the training dataset. The method explores the potential for information leakage in collaborative DL scenarios, where multiple parties contribute data to train a shared DL model. In particular, the paper focuses on the use of GANs in collaborative DL and examines the potential for information leakage through the generator network of the GAN. The authors propose a method to mitigate this information leakage and evaluate its effectiveness through experiments on several datasets. First, each participant trains a local model on its own dataset for several iterations to achieve an accuracy above the pre-set threshold, which is used as a discriminator in the next stage. To train the generator, the weights of the discriminator are first fixed. Then, given a specified class, the generator is learned by producing an image that maximises the classification confidence of the discriminator. The generated image has no explicit correspondence to the training data, and the

2. Background

method is sensitive to the variance of training data [137]. Zhu *et al.* [130] formulated the data recovery task as a gradient regression problem. In DLG, the input image pixel values are treated as random variables optimised using back-propagation. At the same time, the shared model parameters are fixed. The object function measures the Euclidean distance between the shared gradient in FL and the gradient given by the random image input, which is minimised during the training phase. They hypothesised that the optimised input when the model converges is similar to the original training image stored on the local client alone. The experimental results on public benchmark datasets prove the hypothesis is valid, indicating that gradient sharing could lead to privacy data leakage. Zhao *et al.* [138] introduced Improved DLG (iDLG) that addresses the divergence and inconsistent label inference issues of DLG. They found that the derivative value corresponding to the ground-truth label drops in the range of $[-1, 0]$, and other cases lie in the range of $[0, 1]$. It is then feasible to identify the correct label in such a naïve way. In addition to the low accuracy of label inference, DLG based methods often fail to recover the image from the gradient when the variance of the data is large, which is very common for the dataset with a large number of classes. Inverting Gradient (IG) [139] improved the stability of DLG and iDLG by introducing magnitude-invariant cosine similarity measurement for the loss function, namely Cosine Distance (CD). It aims to find images that pursue similar prediction changes from the classification model rather than those that can generate close values with the shared gradient. It shows recognisable results of recovering high-resolution images (*i.e.* 224×224) with a large number of training batches (*i.e.* $\#Batch = 100$). Similar to [139], Jeon *et al.* [140] claimed that only gradient information is not sufficient to reveal private training data. Hence, they introduced GIAS, which applies a pre-trained data-revealing model. Yin *et al.* [141] reported that in an image classification mission, the ground-truth label could be easily revealed from the gradient in the last fully-connected layer, and BN statistics can impressively improve the gradient leakage attack and reveal high-resolution private training images. The generative model-based method is another way of gradient leakage attack. In the work of Generative Gradient Leakage (GGL) [142], they also utilise a GAN to generate the fake data. The weights of GAN are pre-trained and fixed. The trainable parameters in GGL are the input sequence of GAN. The label inference part references from iDLG, so the batch size has to be 1. Unlike others, they use Covariance Matrix Adaptation Evolution Strategy (CMA-ES) as the optimiser to bring less variety to the generated data. The data successfully generated from GGL is not precisely true data, but some similar ones, which benefits GGL strong robustness making it capable of overcoming many defence strategies, such as gradient noising, clipping, or

compression.

Other than gradient leakage attack, neural network poisoning is another common way of model attack. They are two distinct methods of compromising the integrity and performance of ML models, particularly neural networks. Gradient leakage attacks exploit the information shared during the collaborative training process, such as in FL, by maliciously manipulating gradients or extracting sensitive information from the gradient updates. These attacks can lead to privacy exposure problem. On the other hand, neural network poisoning involves injecting carefully crafted malicious data points into the training dataset, which can cause the model to learn incorrect or harmful behaviours. This is typically performed by an adversary with the intent to degrade the model's performance or induce specific vulnerabilities. While both attacks aim to compromise ML models, gradient leakage primarily targets the collaborative training process and focuses on exploiting shared information, whereas neural network poisoning undermines the integrity of the training data directly to influence the model's learning outcomes.

There are several ways to mitigate the effects of gradient leakage, including using larger mini-batch sizes, using techniques such as batch normalisation to reduce the sensitivity of the gradients to the input data, and using techniques such as weight regularisation. Overall, gradient leakage is an important consideration in the design and training of neural network models and understanding its effects and how to mitigate them can be critical for achieving good performance in a wide range of ML tasks.

2.4.4 Gradient Leakage Defence

Gradient leakage defence refers to techniques that aim to prevent the leakage of sensitive information through the gradients of a ML model during training. This is particularly important in scenarios where the model is trained on sensitive data, such as personal or financial data, as the gradients of the model may contain information about the training data that an adversary could exploit. There are several ways in which gradient leakage can occur, which we have discussed above. To defend against gradient leakage, a number of approaches have been proposed. One approach is to use DP optimisation algorithms, which aim to ensure that the gradients of the model do not contain any sensitive information. Another approach is to use perturbation-based defences, which add noise to the gradients of the model to obscure any sensitive information that may be contained within them. Other approaches to gradient leakage defence include the use of MPC algorithms, which allow multiple parties to jointly compute gradients without revealing sensitive information to each other, and the use of HE, which allows computation to

2. Background

be performed on encrypted data without the need to decrypt it. Many efforts have been made to protect private information from the gradient. Most methods, such as gradient perturbation, data obfuscating or sanitising, DP, HE and MPC [115] achieve the protection on the private training data or public shared gradient transmitted between the client and server. Zhu *et al.* [130] experimented with two kinds of noise types, Gaussian and Laplacian, and reported that only the magnitude of distribution variance matters, rather than the noise type. The leakage attack fails when the variance is more significant than 10^{-2} , while the model’s performance degrades dramatically at this level of variance. Chamikara *et al.* [143] proposed a data perturbation method reporting that the technique does not degrade the model performance whilst preserving the privacy of the training data. The input dataset is considered a data matrix and subjected to a new feature space by a multidimensional transformation. They perturbed the input data with various scales of transformation to ensure a sufficient level of perturbation. However, the method relies on a centralised server that controls the generation of global perturbation parameters. Most importantly, the perturbation method may corrupt the architectural information of image-based data. Wei *et al.* [144] utilised DP to add noise to each client’s training dataset and propose a per-example-based DP approach for the client, called Fed-CDP. To enhance the inference performance and level of defence for gradient leakage. However, the experimental results show that although the method prevents the training data from being reconstructed from the gradient, the inference accuracy degrades significantly. On the other hand, the computational cost is expensive since the DP is performed on each training sample. When computing the gradient, both Privacy Enhancing Module (PRECODE) [145] and our proposed FedKL expect to prevent the input information from passing throughout the model. The PRECODE inserts a module ahead of the output layer to translate the latent representation of features by a probabilistic encoder-decoder. The encoder-decoder consists of two fully-connected layers. The first encodes the input features into a sequence and then normalises the sequence based on mean and standard deviation values. The mean value is calculated from the first half of the sequence, while the standard deviation is obtained from the remaining half. At last, the decoder translates the normalised sequence into a latent representation that serves as input to a dynamic decay noise injection method proposed for the output layer. The normalisation process between the encoder and decoder prevents the input information from passing through the gradient. Hence, PRECODE can prevent input information from leaking out from the gradient. Compared to our FedKL, PRECODE inserts two fully-connected layers in front of the outer, which leads to a huge computational cost. That is why only three very shallow DNNs were used for the

experiments in their paper.

Gradient leakage defence is an important issue in the field of ML, and a variety of approaches have been proposed to address it. However, it is still an active area of research, and there is ongoing work to develop more effective and efficient ways of preventing the leakage of sensitive information through gradients.

2.5 Summary

This chapter discusses the background information for the approaches presented in the following chapters. The chapter starts by describing the foundation knowledge of FL, which includes an overview of the classic FedAvg algorithm. This is followed by gradient leakage, which is a security vulnerability in FL systems where the gradients of model updates shared among participating devices are leaked, potentially exposing sensitive data. This can occur when devices have insufficient privacy protection measures or when there are malicious participants in the network. In this chapter, several methods are introduced on the gradient leakage, including DLG, iDLG, IG, GGL, etc. Then I discussed gradient leakage defence. To defend against gradient leakage, several measures can be taken, including the use of MPC algorithms to encrypt and protect the gradients during transmission, the use of DP techniques to add noise to the gradients, and the use of FL algorithms that are specifically designed to prevent gradient leakage.

In the rest of this thesis, I will first introduce one gradient leakage attack method, GRNN, which utilises two branches of generators from GAN to generate fake images and corresponding labels. The method reaches a significant improvement to the DLG and IG. After this, we investigate how to utilise DP and HE to defend the attack. At the same time, we proposed FedBoosting to solve the converge bias problem when deploying FedAvg. In the end, FedKL is introduced for the gradient leakage. We developed a new perspective on the problem, that a key-lock module is inserted into the network to protect private information from being recovered from the leaked gradient.

Chapter 3

Generative Data Leakage Attack

3.1 Introduction

More often than not, the success of DL [39, 81] relies on the availability of a large quantity of data. A centralised learning scheme with a cloud-based distributed computing system is commonly used to speed up the training and scale up to larger datasets [146–150]. However, due to data protection and privacy requirements, such systems are generally infeasible as they require centralised data for training. For instance, data owners (*e.g.* hospital, finance company, or government agent) are not willing or not able to share private data with algorithm and computing platform providers, which causes the so-called “Data Islands” issue. Therefore, decentralised training approaches with data privacy protection are more attractive.

FL [90] was proposed to jointly train a model without directly accessing the private training data. Instead of sharing data, the aggregation server shares a global model and requires the individual data owners to expose the gradient information computed privately only. The gradient is generally considered to be secure to share. However, recent studies found that the gradient-sharing scheme is in fact not privacy-preserved [129, 132, 137, 151]. For example, the presence of a specific property of the training dataset can be identified given the gradient information [137]. Hitaj *et al.* [129] showed the feasibility of generating images that are similar to training images using GAN given any known class label [1]. DLG based models [130, 138] were proposed to approximate the leaked gradient via learning the input while fixing the model weights. However, they are usually unstable and sensitive to the size of the training batch and the resolution of the input image. Furthermore, Geiping *et al.* [139] discussed the theoretical aspect of inverting gradient to its corresponding training data and used magnitude-invariant cosine similarity loss

function in proposed IG, which is capable of recovering high-resolution images (*i.e.* 224×224) with a large number of training batch (*i.e.* $\#Batch = 100$). However, we find that the success rate is relatively low.

GAN is a generative model proposed by Goodfellow *et al.* [1] for image generation. Inspired by the GAN and DLG model, we introduce a gradient-guided image generation strategy that properly addresses the stability and data quality issues of DLG based methods. The proposed GRNN, a novel data leakage attack method is capable of recovering private training images up to a resolution of 256×256 and a batch size of 256. The method is particularly suitable for FL as the local gradient and global model are readily available in the system setting. GRNN consists of two branches for generating fake training data and corresponding labels. It is trained in an end-to-end fashion by approximating the fake gradient that is calculated by the generated data and label to the true gradient given the global model. MSE, WD and TVLoss are used jointly to evaluate the divergence between true and fake gradients. We empirically evaluate the performance of our method on several image classification tasks and comprehensively compared against the state-of-the-art. The experimental results confirm that the proposed method is much more stable and capable of producing images with better quality when a large batch size and resolution are used. The contributions of this work are four-fold:

- We propose a novel method of data leakage attack for FL, which is capable of recovering private training images up to a resolution of 256×256 , a batch size of 256 as well as the corresponding labels from the shared gradient. The implementation of the method is publicly available to ensure its reproducibility.
- We conduct a comprehensive evaluation, where both qualitative and quantitative results are presented to prove the effectiveness of GRNN. We also compare the proposed method against the state-of-the-art, which shows that GRNN is superior in terms of the success rate of attack, the fidelity of recovered data and the accuracy of label inference. In addition, our method is much more stable than others with respect to the size of the training batch and input resolution.
- We conduct a face re-identification experiment that shows using the image generated by the proposed GRNN can achieve higher *Top-1*, *Top-3* and *Top-5* accuracies compared to the state-of-the-art.
- We discuss the potential defence strategies and quantitatively evaluate the effectiveness of our method against noise addition defence strategy.

The rest of the chapter is organised as follows: The proposed method is described in Section 3.2. The details of experimental results, discussions, and potential defence strategies are provided in Section 3.3. Section 3.4 concludes this chapter.

3.2 Proposed Method

There are three major challenges that have not been well addressed by existing data recovery methods as follows: model stability, the feasibility of recovering data from large batch size, and fidelity with high resolution. Both [130, 138, 139] treat the data recovery as a high-dimensional fitting problem driven by the gradient regression objective. The state-of-the-art models are fairly sensitive to the data initialisation, easily fail to capture the individual image characteristic when a large number of gradients are aggregated (*i.e.* FedAvg), and hardly maintain the natural image structures in detail when resolution is increased. In this chapter, we propose the GRNN model which formulates the data recovery task as a data generation problem that is guided by the gradient information. We introduce a GAN model as the image data generator and a simply Fully-Connected (FC) layer as a label data generator, where the so-called fake gradient can then be computed given the shared global model. By jointly optimising both two generators to approximate the true gradient, GRNN can well align the latent space of GAN with the gradient space of the shared global model, furthermore, generate the training data with high fidelity stably. Therefore, GRNN is able to recover the data from the local gradient shared between the client and server in a FL setting, which can also be used for malicious purposes, such as stealing private data from a client. In this section, we will first introduce a neural network model based FL system in Section 3.2.1, then present the proposed GRNN method in Section 3.2.2.

3.2.1 Federated Learning

FL is a distributed collaborative training scheme that consists of multiple clients and one parameter server. The gradients calculated restrictively on client nodes are aggregated on the server node using fusion functions [90, 152]. We use the classic FedAvg method to train DNN over multiple parties, which runs a number of steps of Stochastic Gradient Descent (SGD) in parallel on the client node and then averages the resulting model updates via a central server periodically. The global model is merged by taking the average of gradients from local models according to $\omega' = \sum_i^N \frac{1}{N} \omega_i$, where ω' and ω_i are the gradients of global model and the i_{th} local model, and N is the total number of the clients.

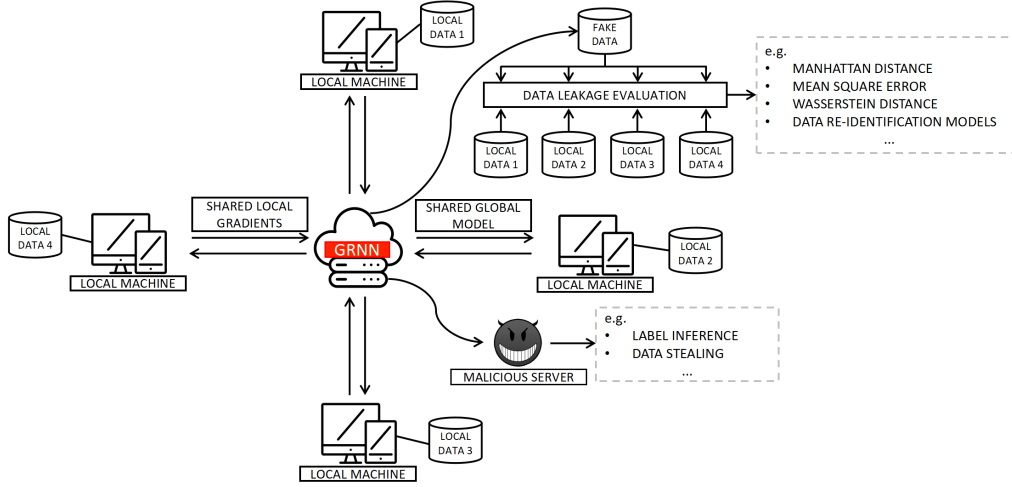


Figure 3.1: Illustration of how GRNN as a malicious server is deployed in a FL system.

We learn a CNN based image classification model, whereas FedAvg is applicable to any finite-sum objective. Formally, at iteration t , the i_{th} ($i \in \{1, 2, \dots, C\}$) client computes the CNN model θ_t and the local gradient g_t^i based on its local training data (x_t^i, y_t^i) (see Equ. 3.1). $\mathcal{F}(\bullet)$, θ_t and $\mathcal{L}(\bullet)$ are the global learning model, network parameters at iteration t and loss function respectively. The local gradient g_t^i is calculated using typical SGD at the client node independently. The server aggregates the local gradient g_t^i and then updates the global model weights θ_{t+1} , as shown in Equ. 3.2 where C is the number of clients.

$$g_t^i = \frac{\partial \mathcal{L}(\mathcal{F}(\langle x_t^i, y_t^i \rangle, \theta_t))}{\partial \theta_t} \quad (3.1)$$

$$\theta_{t+1} = \theta_t - \frac{1}{C} \sum_{i=1}^C g_i \quad (3.2)$$

3.2.2 GRNN: Data Leakage Attack

The proposed GRNN is deployed at the central server as illustrated in Fig. 3.1. The GRNN has access to the neural network architecture and parameters of the global model. The only information that can be obtained between the server and the client is the gradient calculated based on the current global model. To compute the gradient of the model, the private data, corresponding label, and the global model at the current iteration are required at the local client. We hypothesise that the shared gradient contains the information on private data distribution

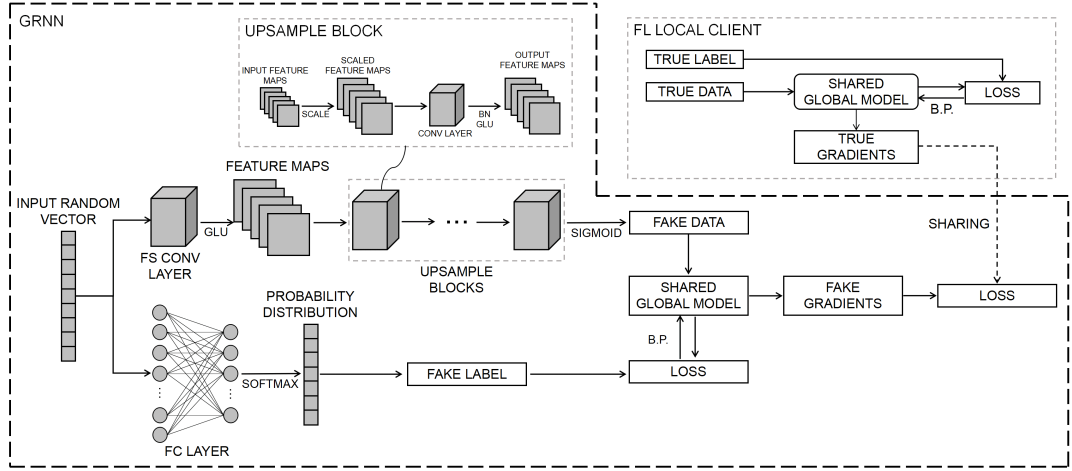


Figure 3.2: Details of the proposed GRNN where the top branch is for generating the fake image and the bottom branch is for inferring the label. “FC LAYER” is the fully-connected layer. “FS CONV LAYER” is the fractionally-strided convolutional layer. For details of upsampling block, please refer to Table 3.1.

and space partitioning given the global model. The architecture of GRNN, shown in Fig. 3.2, has two branches with the same input that is sampled from a common latent space. We consider that each point within this latent space represents a pair of image data and its corresponding label. The objective of the GRNN is to separate this tied representation in the latent space and recover the local image data and corresponding label by approximating the shared gradient that is accessible at the server node.

The top branch is used to recover image data, namely the fake-data generator, which consists of a generative network model. We followed the design principle of iWGAN [153] where the image is generated from a coarse scale to a fine scale gradually. The concept is consistent with a reasonable intuition, where the image is drawn from sketch then the details are gradually added. The input random vector is first fed into a fractionally-strided convolutional layer [154, 155] to produce a set of feature maps with a resolution of 4×4 , which then goes through several upsampling blocks that gradually increase the spatial resolution. The number of the upsampling blocks is determined by the resolution of the target image. For example, if the target image resolution is 32×32 , then 3 upsampling blocks ($4 \rightarrow 8 \rightarrow 16 \rightarrow 32$) are used. We also investigate the large resolution of the input image in Section 3.3.2. In upsampling block, nearest-neighbour interpolation is used to recover the spatial resolution of feature maps from the previous layer. It then passes through a standard convolutional sub-block for rectifying the detail feature representation, which contains a convolutional layer, a BN layer, and a Gated

Table 3.1: Construction of upsampling block.

Layer Name	Setting
Upsampling Layer	scale factor: 2 mode: nearest
Convolutional Layer	kernel size: 3 stride: 1 padding: 1
Batch Normalisation Layer	-
Gated Linear Unit	-

Linear Unit (GLU) [156] activation layer. Empirically, we found GLU is far more stable than *ReLU* and can learn faster than *Sigmoid*. Therefore, we adopted GLU as the activation function for our model while the learning strategy using GAN driven by a regression objective is the key novelty. The proposed method also works with other activation functions, such as *ReLU*. The details of the upsampling block are listed in Table 3.1. At the end of the top branch, a data sample that has the same dimension as the training input of the FL system is generated.

The bottom branch is used to recover label data, namely the fake-label generator, which contains a FC layer followed by a softmax layer for classification. It takes a randomly sampled vector from latent space as input and outputs its corresponding fake label. We assume the elements in the input vector are independent of each other and subject to a standard Gaussian distribution. The label set in GRNN is identical to the one used in the FL system. Formally, the fake image (\hat{x}_t^j) and fake label (\hat{y}_t^j) data generation can be formulated as in Equ. 3.3, where $\hat{\theta}$ and v_t are trainable parameters of GRNN and input random vector is sampled from a unit Gaussian distribution.

$$(\hat{x}_t^j, \hat{y}_t^j) = \mathcal{G}(v_t | \hat{\theta}) \quad (3.3)$$

Given a pair of fake images and labels that are generated as described above, a fake gradient on the current global model can be obtained by feeding them as training input and performing one iteration of SGD according to Equ. 3.1. The objective of GRNN is to approximate the true gradient, therefore, the whole model can be trained by minimising the distance between the fake gradient \hat{g}_t^j and shared true gradient g_t^j , *i.e.* the most commonly used loss MSE is adopted here:

$$\arg \min_{\hat{\theta}} \|g_t^j - \hat{g}_t^j\|^2 \implies \arg \min_{\hat{\theta}} \left\| \frac{\partial \mathcal{L}(\mathcal{F}(\langle x_t^j, y_t^j \rangle, \theta_t))}{\partial \theta_t} - \frac{\partial \mathcal{L}(\mathcal{F}(\langle \hat{x}_t^j, \hat{y}_t^j \rangle, \theta_t))}{\partial \theta_t} \right\|^2$$

Note that the gradient of the model is a vector, the length of which is equal to the number of trainable parameters. In addition to measuring the discrepancy between the true and fake gradients based on Euclidean distance, we also introduce the WD [76] loss to minimise the geometric difference between two gradient vectors and TVLoss [157] to impose the smoothness constrain on generated fake image data. WD, also known as the Earth Mover’s Distance (EMD), is a mathematical metric used to measure the dissimilarity between two probability distributions. It measures the minimum amount of work needed to transform one distribution into another, treating the distributions as masses distributed across space. TVLoss is a regularisation technique to reduce noise and preserve the smoothness of an image. It quantifies the smoothness or variation in an image by calculating the sum of the absolute differences between adjacent pixel values. Therefore, the loss function for GRNN, namely $\hat{\mathcal{L}}(\bullet)$ is formulated as:

$$\hat{\mathcal{L}}(g, \hat{g}, \hat{x}) = MSE(g, \hat{g}) + WD(g, \hat{g}) + \alpha \cdot TVLoss(\hat{x}) \quad (3.4)$$

where we weight the MSE loss and WD equally and α is the weighting parameter for smoothness regularisation. Both branches of the proposed GRNN are parameterised using a neural network that is completely differentiable and can be jointly trained in an end-to-end fashion. The complete training procedure for GRNN is described in Algorithm 1.

Algorithm 1: GRNN: Data Leakage Attack

- 1: $g_t^j \leftarrow \partial \mathcal{L}(\mathcal{F}(\langle x_t^j, y_t^j \rangle, \theta_t)) / \partial \theta_t$; # Produce true gradient on local client.
 - 2: $v_t \leftarrow$ Sampling from $\mathcal{N}(0, 1)$; # initialise random vector inputs for GRNN.
 - 3: **for** each iteration $i \in [1, 2, \dots, I]$ **do**
 - 4: $(\hat{x}_{t,i}^j, \hat{y}_{t,i}^j) \leftarrow \mathcal{G}(v_t | \hat{\theta}_t)$; # Generate fake images and labels.
 - 5: $\hat{g}_{t,i}^j \leftarrow \partial \mathcal{L}(\mathcal{F}(\langle \hat{x}_{t,i}^j, \hat{y}_{t,i}^j \rangle, \theta_t)) / \partial \theta_t$; # Calculate fake gradient on shared global model.
 - 6: $\mathcal{D}_i \leftarrow \hat{\mathcal{L}}(g_t^j, \hat{g}_{t,i}^j, \hat{x}_{t,i}^j)$; # Calculate GRNN loss between true gradient and fake gradient.
 - 7: $\hat{\theta}_{t+1} \leftarrow \hat{\theta}_t - \eta(\partial \mathcal{D}_i / \partial \hat{\theta}_t)$; # Update GRNN model.
 - 8: **end for**
 - 9: **return** $(\hat{x}_{t,I}^j, \hat{y}_{t,I}^j)$; Return generated fake images and labels.
-

3.3 Experiment and Discussion

3.3.1 Dataset and Experimental Setting

A number of experiments on typical computer vision tasks including digit recognition, image classification, and face recognition were conducted to evaluate our proposed method. We used four public benchmarks, MNIST [158], CIFAR-100 [159], Labelled Faces in the Wild (LFW) [160] and VGG-Face [161] for those tasks. There are 7000 grey-scale handwritten digit images of 28*28 resolution in the MNIST dataset. CIFAR-100 consists of 60000 colour images of size 32*32 with 100 categories. LFW is a human face dataset that has 13233 images from 5749 different people. We treated each individual as one class, hence, there are 5749 classes in total. VGG-Face consists of over 2.6 million human face images and 2622 identities.

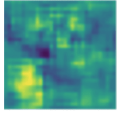
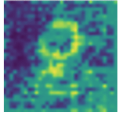
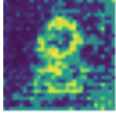



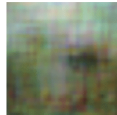

















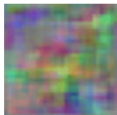




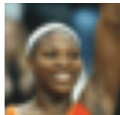
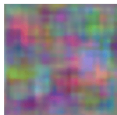
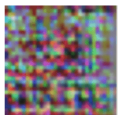

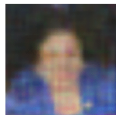
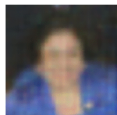
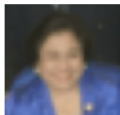
LeNet [162] and *ResNet-18* [39] are used as the backbone networks for training image classifiers in FL system. All neural network models are implemented using PyTorch [163] framework and the source code has been made publicly available¹ for reproducing the results. We replace all *ReLU* functions with *Sigmoid* function in order to ensure the model is second-order differentiable, which is the same as DLG and iDLG in order to intractably compute the MSE loss. Our method is also applicable to the network with *ReLU* function where a second-order differentiable approximation to the *ReLU* function can be used while the computation is intractable compared to using *Sigmoid* function. The batch size varies across different experiments, and a comprehensive comparison study was carried out. *RMSprop* optimiser with a learning rate of 0.0001 and a momentum of 0.99 is used for GRNN. Regarding the loss function of GRNN, we set the weights of TVLoss to $1e - 3$ and $1e - 6$ for *LeNet* and *ResNet-18*, respectively.

3.3.2 Image Recovery

We first trained three CNN based image classifiers over one iteration on MNIST, CIFAR-100, and LFW datasets separately, and then used the proposed GRNN to conduct data leakage attack on those models. In order to demonstrate that our method has no requirement on the convergence of the shared global model and deployment flexibility, we conducted two sets of experiments of the data leakage attack using GRNN. One attack was carried out after the first iteration (non-converged state) and another one was carried out after the change of the loss of shared global model is sufficiently small $\delta G \leq 1e - 4$ (converged state).

¹<https://github.com/Rand2AI/GRNN>

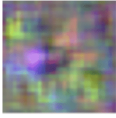

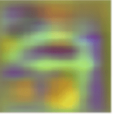



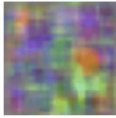





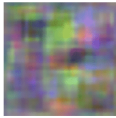
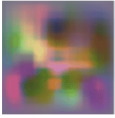




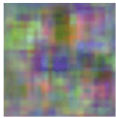





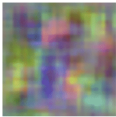





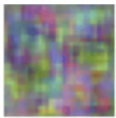
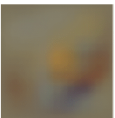
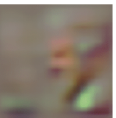



Table 3.2: Examples of data leakage attack using the proposed GRNN on the global model trained over one iteration.

Dataset	Generated Data					True Data
MNIST						
CIFAR-100						
						
						
LFW						
						

Some qualitative results are presented in Table 3.2, where the middle column shows the image data recovered from the true gradient. The images are recovered gradually with the number of iterations for training GRNN increases, while the true gradient is obtained only at the first iteration of the global FL model, which indicates the gradient can lead to data leakage in FL regardless the convergence of shared global model. Table 3.3 shows that image data generated from a converged global model performs worse than that from a non-converged global model. We argue that the reason for lower performance is caused by a large proportion of zero gradients produced by the converged model. We also quantify the quality of generated image using Peak Signal-to-Noise Ratio (PSNR) score, an objective standard for image evaluation which is defined as the logarithm of the ratio of the squared maximum value of RGB image fluctuation over MSE between two images. The formal definition is given as such: $PSNR = 10 \cdot \lg\left(\frac{255^2}{MSE(img1, img2)}\right)$. The higher PSNR score, the higher the similarity between two images. Table 3.4 gives the average

3. Generative Data Leakage Attack

Table 3.3: Examples of data leakage attack using the proposed GRNN on converged global model.

Dataset	Generated Data					True Data
MNIST						
CIFAR-100						
						
						
LFW						
						

PSNR scores achieved by DLG and GRNN with the batch size of 1 using a non-converged and a converged global model. Overall, the non-converged model performs better than the converged model, except using *ResNet-18* on the MNIST dataset, which achieves 1.40 dB less than the converged model (37.27 dB VS. 38.67 dB). DLG failed to recover the training image on MNIST and LFW datasets, as there is no visually recognisable image generated.

To quantitatively evaluate the similarity of recovered images and true images, three metrics, namely Mean Square Error (MSE), Wasserstein Distance (WD) and Peak Signal-to-Noise Ratio (PSNR) [164] are computed. Fig. 3.3 shows the MSE and WD between recovered images and true images with respect to the training iteration of the attacking model. *LeNet* model as the global FL model was used in this experiment. The dash lines and the solid lines correspond to the results of DLG and GRNN, respectively. Although GRNN achieves slightly higher scores in MSE and WD when the batch size of 1 is used, our method is much more stable and significantly

Table 3.4: Average PSNR scores achieved by DLG and GRNN with the batch size of 1 using the non-converged and converged global model. “-” represents that there is no understandable visual image generated.

Method	Model	Dataset	Non-Converged	Converged
DLG	LeNet	MNIST	53.67	-
		CIFAR-100	49.05	33.81
		LFW	41.97	-
Ours	LeNet	MNIST	56.61	30.81
		CIFAR-100	47.03	28.75
		LFW	43.01	28.21
	ResNet-18	MNIST	37.27	38.67
		CIFAR-100	29.57	27.93
		LFW	31.07	27.91

better when a larger batch size is used. When the batch size increases to 4 and 8, DLG only works on MNIST but fails on both CIFAR-100 and LFW, therefore, the corresponding similarity measurements can not converge (see the green and purple dashed lines in Figs. 3.3 (c) (d), (e) and (f)). DLG fails on all datasets with batch size of 16 while GRNN is able to recover the image data consistently (see Figs. 3.3 (g) and (h)). We also notice that DLG can well approximate the shared true gradient while generating a poor image. Fig.3.4 shows that DLG achieves smaller MSE loss (Euclidean distance between true gradient and fake gradient) compared to our method, while it fails to recover the image data.

In Fig. 3.4, MSE and WD results from GRNN are slightly higher than those from DLG with a batch size of 1, however, the difference between these two generated set of images at this batch size is hardly discernible. Hence, we further calculated PSNR to compare the pixel-wise similarity of the recovered images. Table 3.4 shows GRNN achieves higher PSNR on MNIST and LFW datasets (+2.94 dB and +1.04 dB respectively) and lower on CIFAR-100 dataset (-2.01 dB) using the non-converged global model. Furthermore, our method achieved reasonable PSNR score on attacking *ResNet-18* model while DLG always fails. Table 3.5 shows some qualitative comparison of recovered images using both methods over different numbers of iterations. We can observe that DLG recover the image pixel by pixel greedily, whereas GRNN also ensures the appearance distribution to be consistent with the true image in a coarser scale, and object details are then gradually filled at a finer scale.

3. Generative Data Leakage Attack

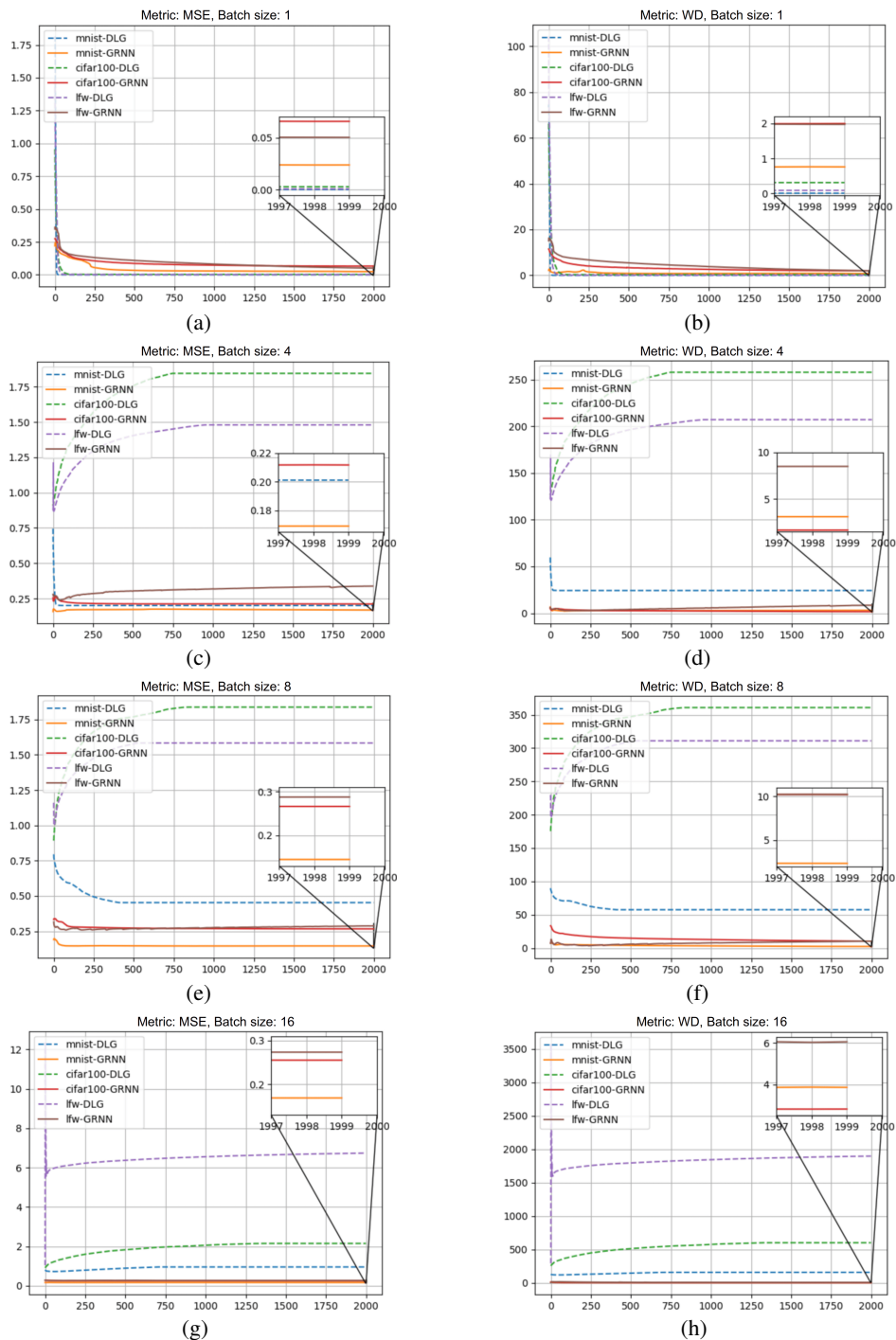


Figure 3.3: Distances between true and generated images with respect to training iteration for DLG and GRNN on three datasets. The horizontal axis corresponds to the number of training iterations of two attacking models and the vertical axis corresponds to the similarity metrics.

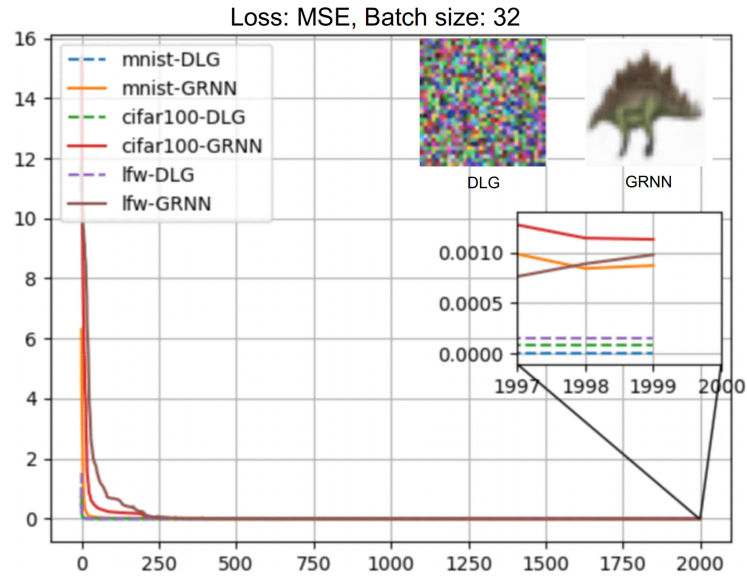





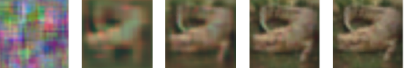







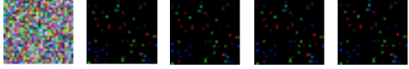



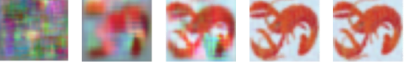


Figure 3.4: MSE loss visualisation for DLG and GRNN with batch size 32.

Table 3.5: Comparison of image recovery using DLG and GRNN over different numbers of iterations.

True Data	DLG	GRNN
	 ✓	 ✓
	 ✓	 ✓
	 ✓	 ✓
	 ✓	 ✓
	 ✗	 ✓
	 ✗	 ✓

3.3.2.1 Study on Batch Size

A comparison study between the proposed GRNN and DLG was carried out using the same setting as described above, while we varied the training batch size for the FL model to evaluate the feasibility of data leakage attack. It is reasonable to consider that the data recovery is more challenging when the training batch size is increasing as the shared gradient is averaged over all image data in the batch, where information of an individual image is obscurely mixed. Table 3.6 lists the success and failure of attack for both two methods. We follow the same principle defined in DLG, where a successful attack refers to recovering an image that is visually recognisable. On the MNIST dataset, DLG attack starts to fail when the size of the training batch is larger than 8 and the *LeNet* is used in FL, whilst our method is able to recover the image data even with a batch size of 256. On CIFAR-100 and LFW datasets, DLG attack only works with a training batch size of 1, however, our method can still successfully perform the attack with a large batch size of up to 64 and 128 respectively. Some failure examples of our method when a large training batch size is used can be found in Table 3.7. Although some failure examples show the consistency of colour distribution and geometric similarity of objects, the appearance details are largely inconsistent or hard to match the original ones. We also show that the proposed GRNN can successfully attack complex and large models, such as *ResNet-18*. Regarding the order of the reconstructed images, when the batch size is 1, it is simple to pair the reconstructed data with the true data. Table 3.2 and Table 3.3 display examples from a batch size of 1. However, when the batch size increases beyond 1, the order of the reconstructed images does not correspond to that of the true images. The gradient is averaged over all images in a batch. GRNN reconstructs the data by matching the averaged local gradient, and the image order does not affect the average gradient’s value.



























3.3.2.2 Study on Success Rate

To quantitatively evaluate the success rate of leakage attack, we conducted a comparison experiment on the testing set of CIFAR-100 using both GRNN and DLG. When the number of batch size is larger than 1, we treat the matching problems between ground-truth images and leaked images as a classic assignment problem given the similarity metrics, such as MSE and PSNR. In addition, Structural Similarity (SSIM) was also introduced to evaluate the structural discrepancy between the two images. Fig. 3.5 shows the success rates of both methods with different numbers of batch sizes against the similarity threshold for the so-called “successful attack”. The lower MSE score indicates the better match, whereas, the higher PSNR and SSIM

Table 3.6: Data leakage attack with different training batch sizes for FL model, where “√” refers to a success and “×” refers to a failure.

Method	Model	Dataset	#Batch							
			1	4	8	16	32	64	128	256
DLG	LeNet	MNIST	√	√	√	×	×	×	×	×
		CIFAR-100	√	×	×	×	×	×	×	×
		LFW	√	×	×	×	×	×	×	×
Ours	LeNet	MNIST	√	√	√	√	√	√	√	√
		CIFAR-100	√	√	√	√	√	√	√	×
		LFW	√	√	√	√	√	√	×	×
	ResNet-18	MNIST	√	√	√	√	×	-	-	-
		CIFAR-100	√	√	×	×	×	-	-	-
		LFW	√	×	×	×	×	-	-	-

Table 3.7: Examples of failed data leakage attack using the proposed GRNN on *LeNet* with batch sizes of 128 and 256. The top row shows the fake images recovered from the shared gradient and the bottom row shows corresponding true images in the private datasets.

Dataset	CIFAR-100				LFW								
Samples													
													

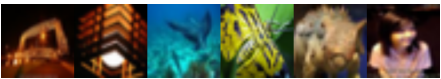

imply the better match. Fig. 3.5 shows that given the same success rate, GRNN achieves significantly lower MSE scores and higher PSNR and SSIM apart from the batch size of 1. For example, with a success rate of 0.6, the MSE threshold is around 0.05 for GRNN and 0.17 for DLG. Similarly, given the same threshold ratio, our method achieves a higher success rate. For instance, when a SSIM threshold of 0.2 is used, most success rates of DLG are dropped down to 0 apart from DLG with a batch size of 1 that achieves around 0.7. However, the worst success rate for GRNN is above 0.77. This experiment can further approve that the proposed method outperforms DLG by a significant margin. More qualitative comparisons that were randomly selected from the generated images are illustrated in Table 3.8.

3.3.2.3 Study on Image Resolution

As aforementioned in Section 3.2.2, GRNN is capable of handling different resolutions of images, due to the flexible number of upsampling blocks. We evaluate the recovery performance

3. Generative Data Leakage Attack

Table 3.8: Some randomly selected ground truth (GT) images and corresponding recovered images from GRNN and DLG with different batch sizes using *LeNet*.

Method	#Batch & Images	
	1	4
GT		
GRNN		
DLG		
	8	16
GT		
GRNN		
DLG		
	32	64
GT		
GRNN		
DLG		
	128	256
GT		
GRNN		
DLG		

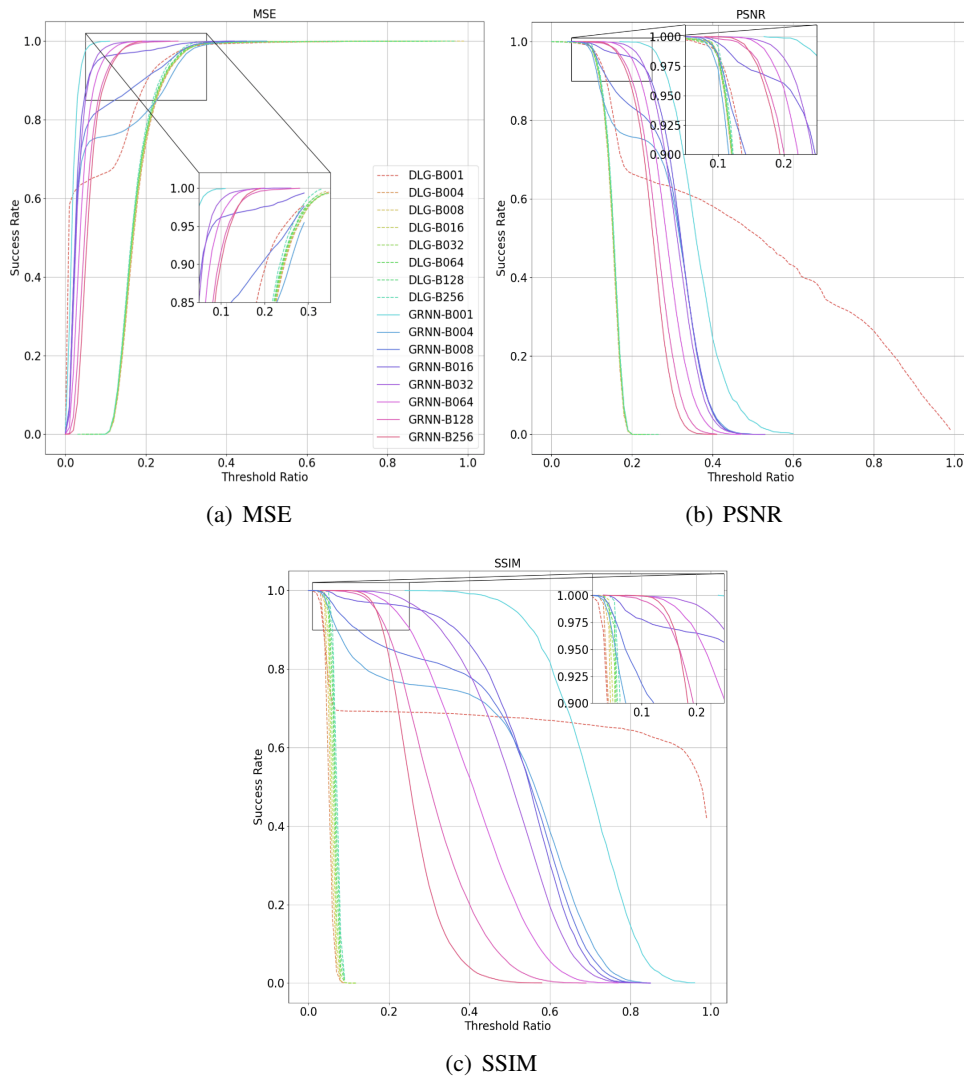


Figure 3.5: Successful attack rate for different batch sizes over normalised threshold ratio on MSE, PSNR and SSIM similarity metrics. The dashed line refers to the results from DLG, and the solid line is from the proposed GRNN

3. Generative Data Leakage Attack

Table 3.9: Data leakage attack using GRNN with different image resolutions and sizes of training batch for FL model, where “✓” refers to a success and “×” refers to a failure. The network is *ResNet-18*, and the dataset is CIFAR-100.

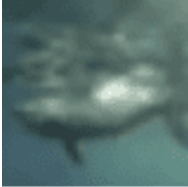

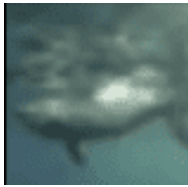






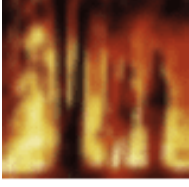

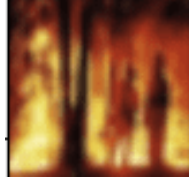
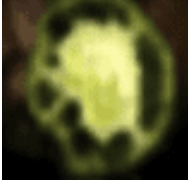



Resolution \ #Batch	1	4	8	16	32
32*32	✓	✓	✓	✓	✓
64*64	✓	✓	✓	×	×
128*128	✓	×	×	×	×
256*256	✓	×	×	×	×

of larger resolutions than 32*32 using *ResNet-18* as the local model and the dataset is CIFAR-100. Table 3.9 shows the performance with different image resolutions and batch sizes. First, we upsample the original image from 32 to 64, the results show that even with the batch size of 8, GRNN is capable of recovering images from the shared gradient. Then we further explore the resolutions of 128*128 and 256*256, both experiments success with a batch size of 1. Some qualitative results in resolutions are shown in Table 3.10. Table 3.11 shows the comparison results of GRNN and IG using a resolution of 256*256, the similarity between the recovered images and original images calculated with MSE, PSNR and SSIM are also given. GRNN outperforms IG by a significant margin for all samples when the SSIM metric is used. It is noticeable that our method is better than IG virtually. Based on this study, we can conclude that our method is also capable of recovering the global structure and colour appearance in the image when a large batch is used, while IG likely produces virtually unrecognisable images.

3.3.2.4 Study on Loss Function

MSE loss is widely used in regression tasks, however, it can be easily biased to the outlier or noisy data point at a pixel-wise level. We believe that the distribution information embedded in the gradient vectors indicates the global structure of the image data. Therefore, we introduced WD distance to measure the geometric discrepancy between the fake gradient and true gradient and guide the image generation process. In addition, we carried out comparison experiments to evaluate different combinations of loss functions including MSE, WD, TVLoss and CD. The results can be found in Table 3.12. The experimental results show that the proposed loss objective combining MSE, WD and TVLoss achieves the best performance. It is noteworthy mentioning that the colour distortion and artefact can be suppressed by using TVLoss and WD jointly as they can effectively penalise the spurious noises locally and globally.

Table 3.10: Recovered images with different resolutions using GRNN. The network is non-converged *ResNet-18*, batch size is 1 and dataset is CIFAR-100.

Resolution	Recovered Images		Ground Truth	
32*32				
64*64				
128*128				
256*256				

3.3.3 Label Inference


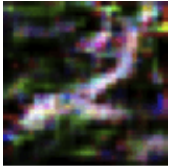








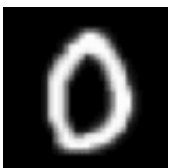
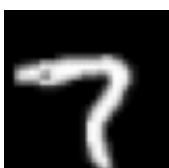




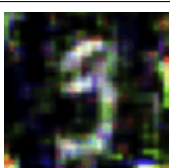


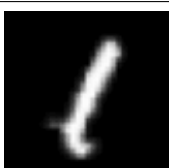


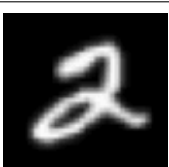
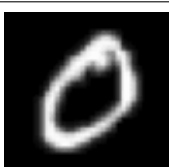
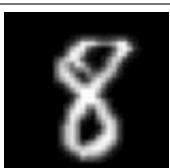

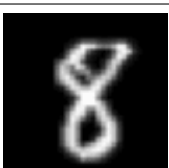

In addition to image data recovery, we also investigated the performance of label inference in those experiments. Table 3.13 provides the comparative results of label inference accuracy (%) for DLG and GRNN, where each experiment was repeated 10 times, and the mean and standard deviation were reported. We can observe that the label inference accuracy decreases while the size of the training batch increases for both DLG and GRNN. However, our method outperforms DLG in all experiments except the one on MNIST with a batch size of 8. Furthermore, GRNN is significantly better than DLG when a large batch size is used. For example, GRNN achieves 99.84% on LFW using *LeNet* and a batch size of 256, whereas DLG only obtains 79.69% using the same setting. Note that having a correct label prediction does not necessarily indicate the corresponding image data can be recovered successfully (see Table 3.6). We can conclude

3. Generative Data Leakage Attack

Table 3.11: Randomly selected images recovered by GRNN and IG. The network is non-converged *ResNet-18*, resolution is 256*256, and dataset is CIFAR-100.

Method	#Batch & Images							
	1							
GT								
	GRNN				IG			
MSE	4107.90	3656.38	45.75	53.21	2272.82	4853.13	94.19	1729.50
PSNR	11.99	12.50	31.52	30.87	14.57	11.27	28.39	15.75
SSIM	0.49	0.90	0.95	0.98	0.31	0.26	0.85	0.55
4								
GT								
	GRNN				IG			
MSE	1482.89	5996.54	1391.17	1799.29	2261.19	4481.52	716.74	2147.60
PSNR	16.42	10.35	16.70	15.58	14.59	11.62	19.58	14.81
SSIM	0.90	0.61	0.75	0.86	0.39	0.34	0.56	0.40
8								
GT								
	GRNN				IG			
MSE	4874.68	1280.33	1441.29	3380.90	4836.14	3309.53	2693.98	3059.81
PSNR	11.25	17.06	16.54	12.84	11.29	12.93	13.83	13.27
SSIM	0.54	0.59	0.70	0.54	0.24	0.28	0.35	0.33

Table 3.12: Comparison of image recovery using different loss functions on MNIST dataset. MSE is Mean Square Error. WD refers to Wasserstein Distance. CD indicates Cosine Distance. And TVLoss is Total Variation Loss

Loss Function	Recovered Images		Ground Truth	
MSE				
WD				
CD				
MSE & WD				
MSE & CD				
MSE & CD & TVLoss				
MSE & WD & TVLoss				

3. Generative Data Leakage Attack






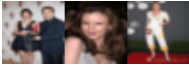




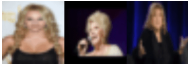
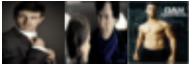
Table 3.13: Comparison of label inference accuracy (%) using DLG and GRNN, where L. and R. refer to LeNet and ResNet-18 respectively.

		#Batch	1	4	8	16	32	64	128	256	Avg
DLG	L.	MNIST	100.0±0.0	97.50±0.0750	100.0±0.0	94.38±0.0337	90.62±0.0242	90.31±0.0260	91.41±0.0152	92.42±0.0181	94.23±0.0254
		CIFAR	100.0±0.0	97.50±0.0750	98.75±0.0375	97.50±0.0306	92.50±0.0287	89.84±0.0329	86.33±0.0295	85.70±0.0289	93.51±0.0329
		LFW	100.0±0.0	80.00±0.3317	90.00±0.1561	85.00±0.2358	88.44±0.1683	70.62±0.3924	89.53±0.2985	79.69±0.3985	85.41±0.2477
Ours	L.	MNIST	100.0±0.0	100.0±0.0	97.50±0.0500	97.50±0.0415	97.50±0.0187	96.25±0.0188	96.25±0.0171	96.37±0.0165	97.73±0.0203
		CIFAR	100.0±0.0	100.0±0.0	100.0±0.0	99.38±0.0188	99.69±0.0094	98.91±0.0122	98.75±0.0080	96.80±0.0108	99.19±0.0074
		LFW	100.0±0.0	97.50±0.0750	98.75±0.0375	100.0±0.0	100.0±0.0	99.69±0.0062	99.69±0.0071	99.84±0.0019	99.43±0.0160
	R.	MNIST	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	-	-	-	100.0±0.0
		CIFAR	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	-	-	-	100.0±0.0
		LFW	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	94.06±0.0452	-	-	-	98.81±0.0090

that recovering image information is much more challenging than recovering labels as the distribution of image data is in a much higher dimension than its corresponding label. The accuracy of label inference on *ResNet-18* achieves 100% in almost all experiments except the one on LFW with a batch size of 32 (94.06%), which is higher and more stable than *LeNet*. *ResNet-18* has much more trainable parameters than *LeNet*, therefore, the gradient with a larger number of elements is much more informative for finding decision boundaries for the classification task.

We also noticed that the number of label classes has an impact on its inference performance. In our experiment, MNIST, CIFAR-100, and LFW have 10 classes, 100 classes, and 5749 classes, respectively. The average accuracy of DLG is 94.23% on MNIST while decreasing to 93.51% and 85.41% on CIFAR-100 and LFW, respectively. In contrast, GRNN achieves higher accuracy on LFW compared to the other two datasets. In DLG, the label is obtained via updating the image input and label input jointly during the backward pass phase in SGD, whereas in GRNN, the label is calculated by the fake label generator in the forward pass phase. We believe that by adding the image data and label generators, GRNN can better capture the correspondence between image data and its label in the joint latent space and, furthermore, can generate more individualised images with respect to different classes.

Table 3.14: The re-identification results of true images and their corresponding generated fake images.

Ground Truth Label	#42	#26	#69
True Input Image			
Top-3 Labels	#42 #97 #94	#26 #22 #64	#22 #64 #23
Top-3 Images			
Top-3 Confidences	40.70% 10.84% 4.08%	90.34% 4.04% 1.75%	66.10% 5.02% 4.71%
Generated Input Image			
Top-3 Labels	#42 #4 #22	#57 #31 #75	#69 #76 #88
Top-3 Images			
Top-3 Confidences	11.89% 7.58% 7.46%	73.10% 7.88% 3.91%	59.52% 7.96% 3.29%

3.3.4 Face Re-Identification

As shown in Table 3.5, recovered images look almost the same as their corresponding true images, and there are still slight deviations that may produce classification misleading results if we use generated data to replace the original ones. This behaviour of DL methods is well documented in the literature, *e.g.* [165–167]. Taking face recognition as an example, the recovered face image may look identical to its original image visually, however, it cannot produce correct prediction by the face recognition model to identify the person, see Table 3.14. Therefore, we applied the face re-identification experiment to evaluate the feasibility of the data leakage attack using GRNN. We first used the GRNN to recover the face image data from the FL system during the training stage. Then, the fake image was passed to the face recognition model as input to predict the identity label. The success of re-identification was counted if the prediction label matched the true label. The VGG-Face dataset was used in this experiment which contains 2622 identities. We used the top 100 identities that have the most image samples for training. We selected the successfully recovered images data from the

3. Generative Data Leakage Attack

Table 3.15: Performances of different network architectures, where training accuracy refers to predicted results of true images and relevant ground truth labels. Re-identification accuracy is from predicted results of fake images and relevant ground truth labels. DLG and GRNN both use *LeNet* as backbone. Training and testing samples are from the VGG-Face dataset. Res18 represents to ResNet-18 and Dense121 is DenseNet-121.

Method	Network	Train Acc	#B	Re-identification Accuracy			Sample No.
				Top-1	Top-3	Top-5	
DLG	Res18	97.27%	1	25.14%	45.57%	51.86%	700
	Dense121	97.11%	1	15.57%	33.57%	42.14%	700
GRNN	Res18	97.27%	1	30.66%	74.79%	88.40%	700
			4	17.45%	24.64%	31.11%	1112
			8	6.14%	13.58%	22.13%	2224
			16	2.90%	10.43%	22.08%	4352
	Dense121	97.11%	1	11.46%	43.12%	63.03%	700
			4	9.53%	19.87%	26.80%	1112
			8	3.06%	10.25%	19.83%	2224
			16	1.52%	8.23%	19.12%	4352

output of GRNN which ended up with 700 fake face images in total when batch size is 1, and 1112 images, 2224 images, and 4352 images for batch size 4, 8, and 16 respectively. In the meantime, we trained two face recognition models using *ResNet-18* and *DenseNet-121* using the same training set. Table 3.15 reports the top-1, top-3 and top-5 accuracies of re-identification of those fake face images. Although the top-1 accuracy on *ResNet-18* and *DenseNet-121* are 30.66% and 11.46% when batch size is 1, they are significantly better than random prediction (1%). The re-identification accuracy increases dramatically when we consider using top-3 and top-5 metrics. The recovery performance becomes worse with the increasing batch size, so the accuracy decreases as well. We say that the face recognition deep models are sensitive to the minor perturbation that is produced during the image recovery process. The images generated using GRNN achieve significantly higher accuracies compared to DLG, *i.e.* +5.52%, +29.22% and +36.54% can be achieved in *Top-1*, *Top-3* and *Top-5* accuracies using *ResNet-18*.

3.3.5 Defence Strategy

The most relevant defence approach for GRNN is noise addition, where, in our scenario, the clients can add a level of Gaussian or Laplacian noise onto the shared gradient. When the gradient is perturbed in this manner, it can mislead the generators into producing poor results since the GRNN relies on matching the gradient to reconstruct accurate data. Essentially, introducing noise into the true gradient can cause the generators to be fooled. We empirically

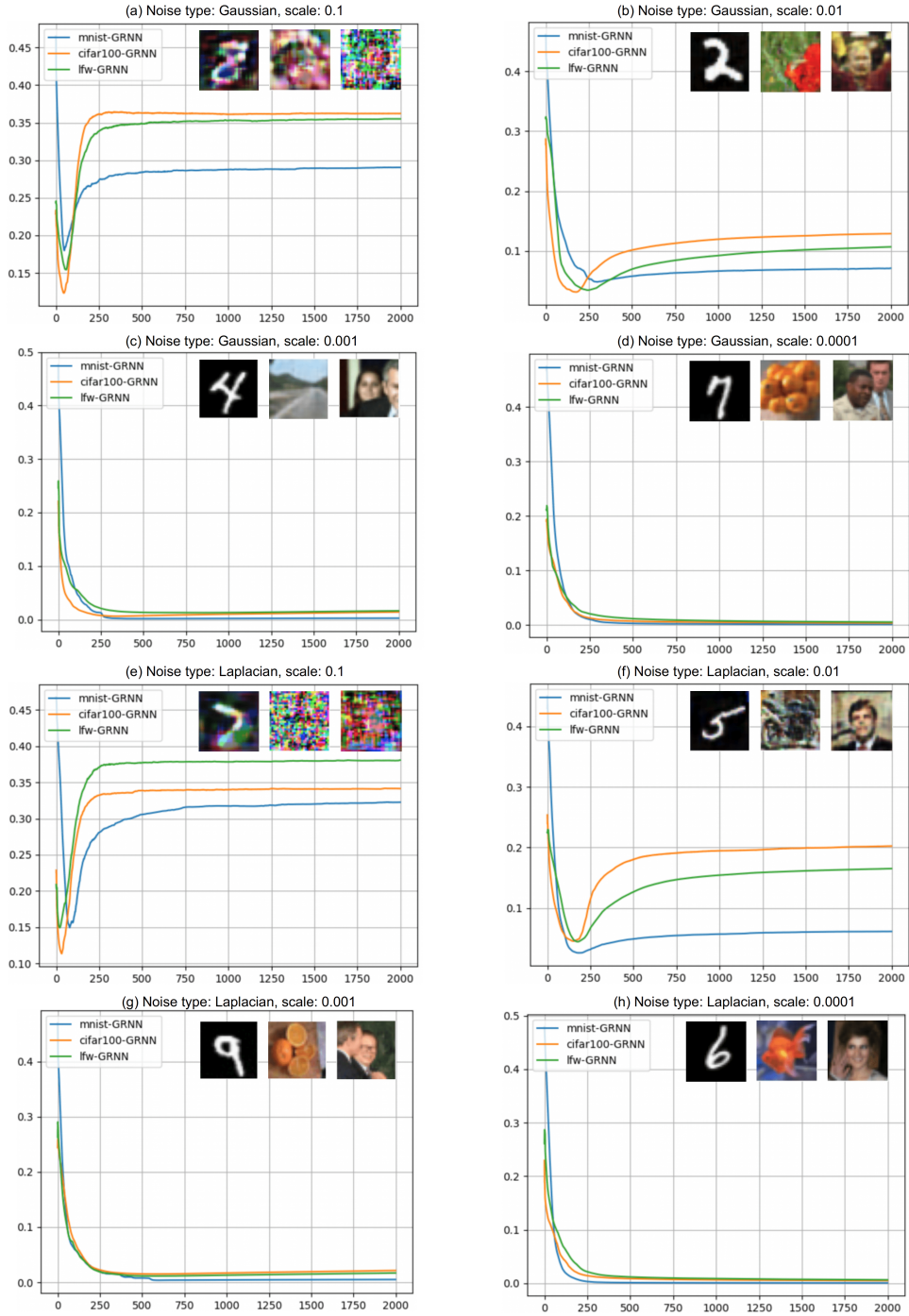


Figure 3.6: MSE distances between true images and generated fake images during training and illustration of generated fake images from GRNN with different noise types and scales on three datasets. (a) - (d) are the results of adding Gaussian noise, and (e) - (h) are the results of adding Laplacian noise. The horizontal axis is the number of iterations for training the attack model.

3. Generative Data Leakage Attack

Table 3.16: Average PSNR scores with different noise types and scales. “×” means the method fails the experiment, whereas PSNR score is given only if it succeed. DLG failed completely, as it had no visible success among all the experiments.

Method	Dataset	Scale		#Batch	1e-1	1e-2	1e-3	1e-4
		Type						
DLG	MNIST	Gaussian	1	×	×	×	×	×
			4	×	×	×	×	×
		Laplacian	1	×	×	×	×	×
			4	×	×	×	×	×
	CIFAR-100	Gaussian	1	×	×	×	×	×
			4	×	×	×	×	×
		Laplacian	1	×	×	×	×	×
			4	×	×	×	×	×
	LFW	Gaussian	1	×	×	×	×	×
			4	×	×	×	×	×
		Laplacian	1	×	×	×	×	×
			4	×	×	×	×	×
Ours	MNIST	Gaussian	1	31.33	31.33	46.27	56.89	
			4	×	31.19	31.21	40.73	
			8	×	32.07	31.17	33.29	
			16	×	×	31.33	31.43	
		Laplacian	1	31.25	30.97	42.03	55.47	
			4	×	31.21	31.53	42.65	
			8	×	×	31.19	33.91	
			16	×	×	31.11	31.87	
	CIFAR-100	Gaussian	1	×	28.23	35.43	43.37	
			4	×	×	28.45	33.15	
			8	×	×	28.23	30.39	
			16	×	×	×	29.17	
		Laplacian	1	×	28.13	32.21	44.27	
			4	×	×	28.15	32.51	
			8	×	×	28.03	30.15	
			16	×	×	×	×	
	LFW	Gaussian	1	×	28.17	35.31	44.39	
			4	×	×	28.37	33.05	
			8	×	×	×	×	
			16	×	×	×	×	
		Laplacian	1	×	28.05	33.79	44.05	
			4	×	×	28.25	32.39	
			8	×	×	×	×	
			16	×	×	×	×	

evaluated the effectiveness of GRNN when the noise addition defence strategy was used. In this study, the *LeNet* was used as the global FL model with different batch sizes. As for the Laplacian mechanism, it adds Laplacian-distributed noise to function f . In this chapter, we set

the l_1 -sensitivity Δf to be 1 and varied ε , which can be defined as: $\lambda = \frac{\Delta f}{\varepsilon} \in [1e-1, 1e-4]$. As for the Gaussian mechanism, it also adds randomness with a normal distribution. Technically, the Gaussian mechanism uses l_2 -sensitivity and parameter δ is counted on. We simplify the Gaussian mechanism to add the noise whose distribution has 0 mean and only ranges standard deviation from $1e-1$ to $1e-4$. Fig. 3.6 shows the MSE distance between the recovered image and the true image when different levels and types of noises are added. GRNN fails to recover the image when a high level of noise is added to the gradient (see Figs. 3.6 (a) and (e)), however, this usually leads to poor performance on the global FL model as the noisy gradients are aggregated. We observed that GRNN is able to recover the image data successfully and obtains reasonable results when the scale of noise is reduced to 0.01 (see Figs. 3.6 (b) and (f)). The average PSNR scores are presented in Table 3.16. Compared to the no-defence approach shown in Table 3.4, the noise added to the gradient can result in decreasing of PSNR of the generated images, which indicates the effectiveness of the noise addition strategy. However, the proposed GRNN is still capable of recovering image data when a high level of noise is added to the gradient, *i.e.* $1e-2$. The PSNR scores with a Gaussian noise scale of $1e-4$ on the MNIST dataset are even larger than that without noise (56.89 dB VS. 56.61 dB), as well on LFW dataset (44.39 dB VS. 43.01 dB). On the other hand, we found that even if the image recovery fails, the label can still be inferred correctly using our GRNN. However, DLG totally fails to recover images and inference labels in all of our experiments.

3.4 Summary

In this chapter, we present a data leakage attack method, namely GRNN, for FL system which is capable of recovering both data and its corresponding label. Compared to the state-of-the-art methods, the proposed method is much more stable when a large resolution and batch size are used. It also outperforms the state-of-the-art in terms of fidelity of recovered data and accuracy of label inference. Meanwhile, the experimental results on the face re-identification task suggest that GRNN outperforms DLG by a margin in terms of *Top-1*, *Top-3* and *Top-5* accuracies. We also discussed the potential defence strategies and empirically evaluated the performance of GRNN when noise addition defence is applied. We conclude that our method can successfully and consistently recover the data in FL when a high level of noise is added to the gradient. The implementation of our method is publicly available to ensure its reproducibility. In the rest of the thesis, we will talk about two ways of gradient leakage defence.

Chapter 4

Protected Gradient Boosting

4.1 Introduction

Personal data protection and privacy-preserved issues have particularly attracted researchers' attention [122, 168–173]. Typical ML approaches that require centralised data for model training may not be possible as restrictions on data sharing are in place. Therefore, decentralised data-training approaches are more attractive since they offer desired benefits in privacy preservation and data security protection. FL [90, 92] was proposed to address such concerns that allows individual data providers to collaboratively train a shared global model without aggregating the data centrally. McMahan *et al.* [90] presented a practical decentralised training method for deep networks based on averaging aggregation. Experimental studies were carried out on various datasets and architectures, which demonstrated the robustness of FL on unbalanced and IID data. Frequent updating approach can generally lead to higher prediction performance whereas the communication cost increases sharply, especially for the large datasets [90, 91, 174–176]. Konečný *et al.* [91] focused on addressing the efficiency issue and proposed two weight updating methods, namely structured updates and sketched updates approaches based on FedAvg to reduce the up-link communication costs of transmitting the gradients from the local machine to the centralised server.

Prediction performance and data privacy are two major challenges in FL research. On one hand, the accuracy of FL reduces significantly on Non-IID data [177]. Zhao *et al.* [177] showed the weight divergence can be measured quantitatively using EMD between the distributions over classes on each local machine and the global population distribution. Hence, they proposed to share a small subset of data among all the edge devices to improve model generalisation on

Non-IID data. However, such a strategy is infeasible when restrictions on data sharing are in place which usually leads to privacy breaching. Li *et al.* [178] studied the convergence properties of FedAvg and concluded a trade-off between its communication efficiency and convergence rate existed. They argued that the model converges slowly on heterogeneous datasets. Based on our empirical study in this chapter, we confirm that given Non-IID datasets, the training needs far more iterations to reach an optimal solution and often fails to converge, especially when the local models are trained on large-scale datasets with a small number of batch size or the global model are aggregated after a large number of epochs. On the other hand, the model gradient is generally considered to be safe to share in the FL system for model aggregation. However, some studies have shown that it is feasible to recover training data information from model gradients. For example, Fredrikson *et al.* [151] and Melis *et al.* [137] reported two methods that can identify a sample with certain properties is in the training batch. Hitaj *et al.* [129] proposed a GANs model as an adversarial client to estimate the distribution of the data from the output of other clients without knowing their training data. Zhu *et al.* [130] and Zhao *et al.* [138] demonstrated data recovery can be formulated as a gradient regression problem assuming the gradient from a targeted client is available, which is a largely valid assumption in most FL systems. Furthermore, GRNN proposed by Ren *et al.* [131] consists of two branches of generative models, one is based on GAN for generating fake training data and the other one is based on the fully-connected layer for generating corresponding labels. The training data is revealed by regressing the true gradient and the fake gradient generated by the fake data and relevant label.

In this chapter, we propose the FedBoosting method to address the weight divergence and gradient leakage issues in the general FL framework. Instead of treating individual local models equally when the global model is aggregated, we consider the data diversity of local clients in terms of the status of convergence and the ability of generalisation. To address the potential risk of data leakage via shared gradients, a DP based linear aggregation method is proposed using HE [2] to encrypt the gradients, which provides two layers of protection. The proposed encryption scheme only leads to a negligible increase in computational cost.

The proposed method is evaluated using a text recognition task on public benchmarks, as well as a binary classification task on two datasets, which demonstrates its superiority in terms of convergence speed, prediction accuracy and security. The performance reduction due to encryption is also evaluated. Our contributions are three-fold:

- We propose a novel aggregation strategy namely FedBoosting for FL to address the weight

divergence and gradient leakage issues. We empirically demonstrate that FedBoosting converges significantly faster than FedAvg while the communication cost is identical to traditional approaches. Especially when the local models are trained with a small batch size and the global model are aggregated after a large number of epochs, our approach can still converge to a reasonable optimum whereas FedAvg often fails in such case. Our implementation of proposed FedBoosting is publicly available to ensure reproducibility. It can also be run in a distributed multiple GPUs setup.¹

- We introduce a dual layer protection scheme using HE and DP to encrypt gradients flowing between server and clients, which protect the data privacy from gradient leakage attack.
- We show the feasibility of our method on two datasets by evaluating the decision boundaries visually. Furthermore, we also demonstrate its superior performance in a visual text recognition task on multiple large-scale Non-IID datasets compared to the centralised approach and FedAvg. The experimental results confirm that our approach outperforms FedAvg in terms of convergence speed and prediction accuracy. It suggests FedBoosting strategy can be integrated with other DL models in the privacy-preserving scenarios.

The rest of the chapter is organised as follows: The proposed method FedBoosting and relevant encryption method are described in Section 4.2 and evaluated on a text recognition task and a binary classification task. The details of experiments and discussions on the results, as well as a performance comparison, are provided in Section 4.3, followed by the conclusions in Section 4.4.

4.2 Proposed Method

4.2.1 FedBoosting Framework

FedAvg [90] produces a new model by averaging the gradients from local clients. However, on the Non-IID data, the weights of local models may converge in different directions due to inconsistency of data distribution. Therefore, a simple averaging scheme performs poorly, especially when strong bias and extreme outlier exist [177–179]. Boosting is a powerful ensemble learning technique used to improve the performance of ML algorithms. It involves

¹<https://github.com/Rand2AI/FedBoosting>

4. Protected Gradient Boosting

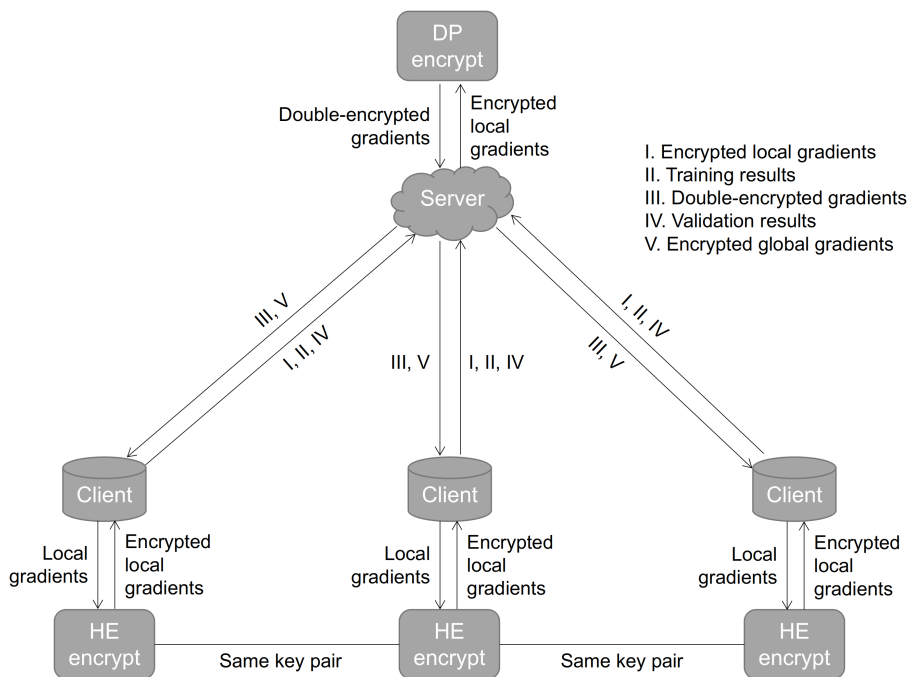


Figure 4.1: The schematic diagram of proposed FedBoosting and encryption protocol. There are two clients used for demonstration purposes, whereas the proposed method can work with an arbitrary number of local clients.

training a sequence of weak learners. The idea is to combine these weak learners to create a strong, more accurate model. This process continues for multiple rounds, with the final model being a combination of the weak learners' predictions. Adaptive Boosting (AdaBoost) [180] focuses on re-weighting misclassified data points in each iteration to prioritise them in the next round of training. Gradient Boosting Machines (GBM) [181] focuses on minimising the residual errors by training weak learners on the negative gradient of the loss function. eXtreme Gradient Boosting (XGBoost) [182] is an optimised and parallelised implementation of gradient boosting that offers several advantages, such as regularisation, improved scalability, and handling of missing data.

We propose using boosting scheme, namely FedBoosting, to adaptively merge local models according to the generalisation performance on different local validation datasets. Meanwhile, in order to preserve data privacy, information exchanges among decentralised clients and server are prohibited. Hence, instead of exchanging data between clients, encrypted local models are exchanged via the centralised server and validated on each client independently. More details

are shown in Figure 4.1.

In contrast to FedAvg, the proposed FedBoosting takes the fitness and generalisation performance of each client model into account and adaptively merges the global model using different weights on all client models. To achieve this, three different pieces of information are generated from each client, local gradients G_r^i , training loss T_r^i and validation loss $V_r^{i,i}$, where G_r^i and T_r^i are local gradients and training loss from the i -th local model in training round r and $V_r^{i,i}$ refers to validation loss from the i -th local model on the i -th local validation dataset in training round r . The local gradients G_r^i are then distributed to all the other clients via a centralised server. The cross-validated loss $V_r^{i,j}$, where $i \neq j$, can be gained on each client. Training and validation losses are two measurements used to evaluate the predictive performance of the models. It is a valid assumption that a model with relatively large training loss indicates poor convergence and poor generalisation ability. However, it also suggests the model gradient contains sufficient information for training. Similarly, a model with low training loss does not guarantee the model having good generalisation ability (over-fitting) with less training information. Hence, we take validation loss into consideration. Those two losses jointly determine the aggregation weight of a local model contributing to the global model as shown in Eqn. (4.2). Then on the server, all the validation results of the i -th model are added together denoted as V_r^i representing the i -th model's generalisation ability. Considering the convergence, a softmax layer is deployed, whose input is T_r . The outputs together with V_r^i are used for calculating the aggregation weight p_r^i . In the current round of aggregation, the new global gradients G_r can be computed by merging all the local gradients G_r^i with respect to its weight p_r^i as such:

$$G_r = \sum_i^N p_r^i G_r^i; \forall p_r^i \in [0, 1], \quad (4.1)$$

$$p_r = \text{softmax}(\text{softmax}(T_r) \cdot \sum_j^N V_r^j), \quad (4.2)$$

$$V_r^{i,j} = \begin{pmatrix} V_r^{1,1} & V_r^{1,2} & \dots & V_r^{1,j} \\ V_r^{2,1} & V_r^{2,2} & \dots & V_r^{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ V_r^{i,1} & V_r^{i,2} & \dots & V_r^{i,j} \end{pmatrix} \quad (4.3)$$

where T_r^i and p_r^i are training loss and mixture coefficient for the i -th local model.

In addition, the proposed FedBoosting scheme is resilient to some malicious attacks, such as data poisoning. For instance, when a malicious client injects poisoned data into the training set and the contaminated local model is aggregated with the same weight as other clean models,

4. Protected Gradient Boosting

our method can mitigate as the validation scores of the toxic model on other clients will be significantly lower, which in turn leads to a significantly lower aggregation weight.

Algorithm 2: FedBoosting with HE and DP: Server

```

1: build model and initialise weights  $\omega_0$ ;
2: for each round  $r = 1, 2, \dots, R$  do
3:   for each client  $i \in C_N$  do
4:     if  $r == 1$  then
5:        $g_r^{*i}, T_r^i \leftarrow \text{Train}(r, i, \omega_0)$  via Algorithm 1.a;
6:     else
7:        $g_r^{*i}, T_r^i \leftarrow \text{Train}(r, i, G_{r-1}^*)$  via Algorithm 1.a;
8:     end if
9:   end for
10:  for each client  $i \in C_N$  do
11:    generate  $\hat{G}_r^{*i}$  via Eqn. (4.5);
12:     $V_r^i \leftarrow \sum_j^N \text{Evaluate}(j, \hat{G}_r^{*i})$  via Algorithm 1.b;
13:  end for
14:  generate  $p_r^i$  via Eqn. (4.2&4.3);
15:  generate  $G_r^*$  via Eqn. (4.4);
16:  if  $r == R$  then
17:     $\omega_r \leftarrow \text{Decrypt}(G_r^*)$  via Algorithm 1.c
18:  end if
19: end for

```

4.2.2 HE Aggregation with Quantized Gradient

HE ensures that the computation can be carried out on the encrypted data as $\text{Enc}(A) \bullet \text{Enc}(B) = \text{Enc}(A * B)$, where “ \bullet ” stands for operation on encrypted data and “ $*$ ” is an operation on plain data. Since the FedBoosting involves computing the global model based on local gradient, the HE is used in our method to ensure the aggregation and gradient exchange among clients and server are secure. In FedBoosting, local models are trained on each client and then local gradients are transmitted to the server, where all local gradients are integrated to build global gradients in every round of aggregation. To preserve gradient information, FedBoosting utilises the HE method, *Paillier* [2]. Once the training starts, a pair of HE keys are shared among clients, but keep secured to the parameter server. The public key is used to encrypt gradients and the secret key is for decryption. After a round of local training, local gradients can be calculated by $G_r^i = \omega_r^i - \omega_{r-1}$, where ω_{r-1} is the global weight from the last round and ω_r^i is the weight after training at current round.

Algorithm 3: FedBoosting with HE and DP: Client

```

1 a. Train( $r, i, \omega_{r-1} \parallel G_{r-1}^*$ ):
  1: if  $i == 1$  then
  2:   generate key pair and sent to other clients
  3: else
  4:   wait for key pair from  $C_1$ 
  5: end if
  6: if  $r == 1$  then
  7:   load  $TrnD^i, ValD^i$ 
  8: else
  9:   decrypt  $G_{r-1}^*$  to  $G_{r-1}$  by secret keys
 10:    $\omega_{r-1} = \omega_{r-2} + G_{r-1}$ 
 11: end if
 12: for each epoch  $e = 1, 2, \dots, E$  do
 13:   for each batch  $b = 1, 2, \dots, B$  do
 14:      $\omega_r \leftarrow \omega_{r-1} - \eta l(TrnD^{i,b}, \omega_{r-1})$ 
 15:   end for
 16: end for
 17:  $G_r^i = \omega_r - \omega_{r-1}$ 
 18:  $g_r^i = \lfloor (G_r^i * 1e^{32}) / P \rfloor$  and generate  $g_r^{*i}$  by public keys
 19:  $T_r^i \leftarrow f(TrnD^i | \omega_r)$ 
 20: return  $g_r^{*i}, T_r^i$  to server

b. Evaluate( $j, \hat{G}_r^{*i}$ ):
  1: decrypt  $\hat{G}_r^{*i}$  to  $\hat{G}_r^i$  by secret keys
  2:  $\hat{\omega}_r^i = \omega_{r-1} + \hat{G}_r^i$ 
  3:  $V_r^{i,j} \leftarrow f(ValD^j | \hat{\omega}_r^i)$ 
  4: return  $V_r^{i,j}$  to server

c. Decrypt( $G_r^*$ ):
  1: decrypt  $G_r^*$  to  $G_r$  by secret keys
  2:  $\omega_r = \omega_{r-1} + G_r$ 
  3: return  $\omega_r$  to server

```

It is infeasible to encrypt G_r^i and to transmit it to the server directly, as *Paillier* can only process integer values. To address this issue, we propose to convert G_r^i into scaled integer form, denoted as G_r^i by multiplying with $1e^{32}$. As the weighting scheme at the server side will break the integer-only constrain for homomorphic computation, to ensure the correctness of aggregation, we divide G_r^i into P pieces and then round to an integer according to $g_r^i = \lfloor G_r^i / P \rfloor$. There is a negligible precision loss as only the last few bits are dropped. For example, in the case of $P = 10$, the value loss is only at the 32-th bit, similarly, for $P = 100$, the loss is at the 31-th

and 32-th bits. Finally, g_r^i is encrypted using *Paillier* and the encrypted g_r^{*i} is transmitted to the server. On the contrary, the client weight p_r is converted to an integer by multiplying P followed by a rounding operation. In FedBoosting, the aggregation weight is computed with respect to Eqn. (4.2 & 4.3). The final encrypted global gradients G_r^* can be computed by merging all gradients from clients (see Eqn. (4.4)). The final encrypted global gradients G_r^* are then transmitted back to each client to be decrypted and generate global weights by $\omega_r = \omega_{r-1} + G_r$, where G_r is decrypted from G_r^* . The proposed secure aggregation approach using HE with quantized gradient is generalised that can also be used for FedAvg.

$$G_r^* = \sum_i^N \lfloor p_r^i \cdot P \rfloor \cdot g_r^{*i}; \forall p_r^i \in [0, 1] \quad (4.4)$$

4.2.3 DP Fusion for Local Model Protection

The local gradient between client and server is protected by HE as aforementioned. While in our proposed FedBoosting mechanism, local models are shared with all the other clients for cross-validation, and all clients have the same key pair. FedAvg shows that the uniformly combined global model is capable of performing similarly as any local model. Therefore, to protect gradient privacy among clients, inspired by DP, we propose to perturb individual local models using a linear combination of HE-encrypted local models, where the target model takes the dominant proposition giving the highest weight. Only the perturbed local models are shared among clients for cross-validation. Empirically, the reconstructed model performs similarly to the local model. Once the server receives all the encrypted local gradients piece $g_r^{*i}, \forall i = 1, 2, \dots, N$, the server randomly generates N sets of private fusing weights within which the corresponding local model always takes the largest proposition. Then, the server computes N reconstructed local model using the HE according to the N sets of weights (see Eqn. (4.5)).

$$\hat{G}_r^{*i} = \lfloor \hat{p} \cdot P \rfloor \cdot g_r^{*i} + \sum_{j:j \neq i}^N \lfloor \frac{(1 - \hat{p}) \cdot P}{N - 1} \rfloor \cdot g_r^{*j} \quad (4.5)$$

where \hat{G}_r^{*i} is the i -th dual-encrypted whole gradient in round r . Finally, the server distributes the reconstructed local models to all clients for cross-validation. As HE is used on the server for linear combination, the model is restrictively protected during the exchange process between the server and clients. The accuracy might drop due to precision loss using quantized HE and reconstructed local model for cross-validation. However, in Figure 4.5, our experimental results show that there is no significant (within 0.5%) loss in testing accuracy.

4.3 Experiment and Discussion

4.3.1 Decision Boundary Comparison using Synthetic Dataset

We first conducted the evaluations using two datasets to compare the decision boundary between FedAvg and FedBoosting. The task is a binary classification problem with the 2D feature in order to provide a visible visualisation for the decision boundary. We assume the data is subjected to a 2D Gaussian distribution and two datasets were randomly sampled with different mean centres and stand deviations in order to simulate the Non-IID scenario. The individual dataset was used for training on a client and the global model was aggregated using FedAvg and proposed FedBoosting, where each of them contains 40000 samples and was split into a training set and a testing set by a proportion of 9:1. Figure 4.2 (d), (e), and (f) show those two training datasets and the combined testing dataset respectively. A simple neural network was adopted that contains 2 fully-connected layers followed by a *Sigmoid* activation layer and a *Softmax* layer respectively. The first fully-connected layer has 8 hidden nodes and the second one has 2 hidden nodes. The optimiser is *Adam* whose learning rate is 0.003. All the models trained by FedBoosting outperform those trained by FedAvg with a batch size of 8 and epoch of 1. Figure 4.2 (a) and (b) present the visualisations of decision boundary of global models trained using FedAvg, FedBoosting and a centralised training scheme respectively. It can be concluded that the proposed FedBoosting can form a significantly smoother decision boundary compared to the FedAvg approach. In addition, the decision boundary of our method is much closer to the model that was trained using a centralised scheme, where both two datasets are used together. This study shows that our method is more generalised in principle.

4.3.2 Evaluation on Text Recognition Task

We adopt Convolutional Recurrent Neural Network (CRNN) [183] as the local neural network model for the text recognition mission. CRNN uses *VGGNet* [37] as the backbone network for feature extraction, where the fully-connected layers are removed and the feature maps are unrolled along the horizontal axis. To model the sequential representation, a multi-layer Bidirectional Long-Short Term Memory (BiLSTM) network [184] is placed on the top of the convolutional layers that take the unrolled visual features as input and models the long-term dependencies within the sequence in both directions. The outputs of BiLSTM are fed into a *Softmax* layer, and each element unrolled sequence is projected to the probability distribution of possible characters. The character with the highest *Softmax* score is treated as an intermediate

4. Protected Gradient Boosting

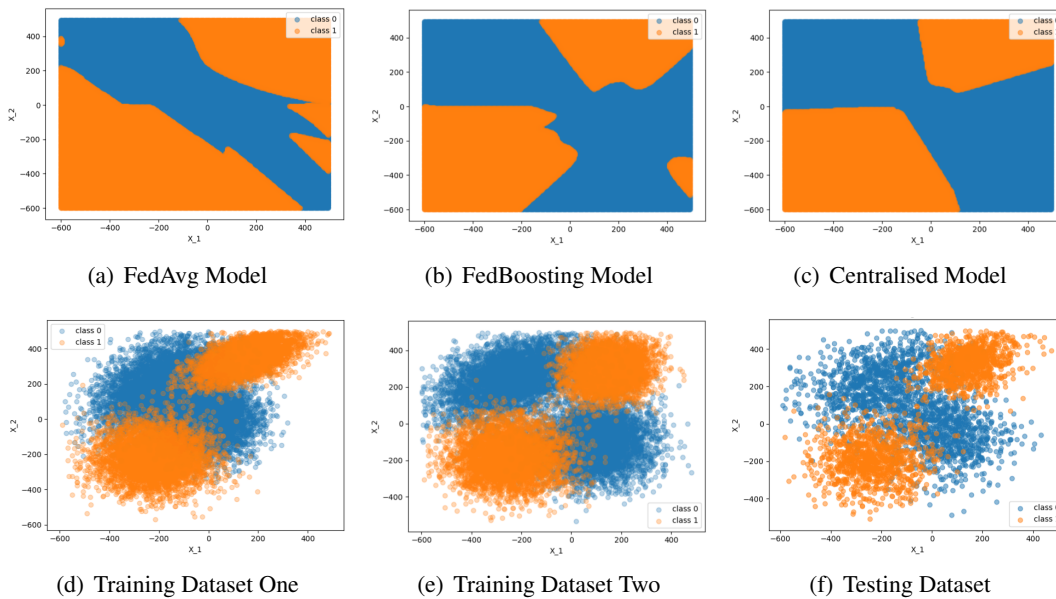


Figure 4.2: The figures (a), (b) and (c) in the first row show the decision boundaries of global models trained using FedAvg, FedBoosting and non FL method (centralised training scheme). Figures (d), (e) and (f) in the second row show two training datasets for two clients in a FL setting and the testing dataset. The model obtained in figure (c) is trained using all datasets, including (d) and (e).

prediction. Connectionist Temporal Classification (CTC) [185] decoder is utilised to merge intermediate prediction to produce the final output text. For more details of CRNN model, the reader can refer to its original publication [183].

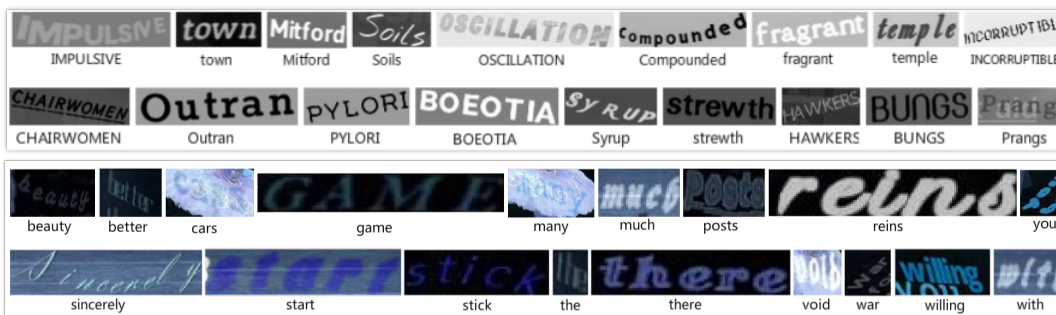


Figure 4.3: Visual example of training images taken from *Synth90K* (top two rows) and *SynthText* (bottom two rows) datasets.

Table 4.1: Recognition accuracies (%) on four testing datasets. “90K” and “ST” stand for *Synth90K* and *SynthText* datasets respectively. The results of the first row (CRNN*) and the second row (CRNN) are produced by the normal CRNN model without using FL framework, where CRNN* corresponds to accuracies reported in [183] and CRNN corresponds to the results reproduced using our implementation.

Method	Dataset	#Batch	#Epoch	IIIT5K	SVT	SCUT	IC15
CRNN*	90K	-	-	81.20	82.70	-	-
CRNN	ST	512	-	76.07	77.60	89.38	55.92
	ST	800	-	73.69	78.00	86.94	58.88
	90K	512	-	80.95	86.40	87.49	67.43
	90K	800	-	80.71	83.60	87.80	62.50
	ST & 90K	512	-	83.81	90.40	93.08	71.71
	ST & 90K	800	-	85.48	88.00	93.78	72.70
FedAvg	ST & 90K	256	1	-	-	-	-
	ST & 90K	256	3	-	-	-	-
	ST & 90K	512	1	85.48	87.60	93.31	73.36
	ST & 90K	512	3	80.83	87.60	91.11	64.80
	ST & 90K	800	1	86.67	89.60	94.49	72.37
	ST & 90K	800	3	81.82	88.00	93.47	70.07
Ours	ST & 90K	256	1	85.83	89.2	94.26	72.37
	ST & 90K	256	3	84.88	91.20	94.65	70.07
	ST & 90K	512	1	85.12	88.00	93.39	74.34
	ST & 90K	512	3	87.38	90.80	93.86	70.39
	ST & 90K	800	1	87.62	89.20	94.18	75.99
	ST & 90K	800	3	85.60	91.60	94.65	70.72

4.3.2.1 Experimental Setting

The proposed model is trained on two large-scale synthetic datasets, *Synthetic Text 90k* and *Synthetic Text*, without fine-tuning on other datasets. Models are tested on other four standard datasets to evaluate their general recognition performances. For all experiments, the prediction performance is measured using word-level accuracy.

Synthetic Text 90k [186] (Synth90K) is one of two training datasets for all the experiments in this chapter. The dataset contains about 7.2 million images and their corresponding ground truth words. We split the images into two sets for FedBoosting, the first one containing 6.5 million images is for training, and the rest is for validation which contains 0.7 million images.

Synthetic Text [186] (SynthText) is the second training dataset we used. The dataset has about 85K natural images containing many synthetic texts. We cropped all the texts via labelled text bounding boxes to build a new dataset of 5.3 million text images. We split it into a training dataset of 4.8 million images and a validation dataset of 0.5 million images for FedBoosting.

IIIT 5K-Words [187] (IIIT5K) is collected from the internet containing 3000 cropped word

4. Protected Gradient Boosting

Table 4.2: Visual example of testing results using FedAvg and FedBoosting models with a batch size of 512 and epochs of 3. Incorrectly predicted characters are highlighted in red, and green characters in brackets are the ones missing from the predictions.

Samples	FedAvg	FedBoosting
	MOUNTRIN	MOUNTAIN
	STATIONNN	STATION
	PLAZZA	PIAZZA
	VIRG(IN)	VIRGIN
	MAXIVUS	MAXIMUS
	UGREENN	GREEN
	JECOIL	ECOIL
	DIR(E)CION	DIRECTORY
	CENIRAL	CENTRAL
	CANSY	CANDY
	ABARTMENTS	APARTMENTS
	WORKSHOPH(E)	WORKSHOPE
	TISC	TSC
	CRIAVEN	CRAVEN
	20MBIES	ZOMBIES
	344	844
	17040	1040
	1800GOG EICO	1800G0GE1CO

images in its testing set. Each image contains a ground truth word.

Street View Text [188] (SVT) is collected from the *Google Street View*, consists of 647 word images in its testing set. Many images are severely corrupted by noise and blur, or have very low resolutions.

SCUT-FORU [189] (SCUT) consists of 813 training images and 349 testing images. The background and illumination vary on large scales in the dataset.

ICDAR 2015 [190] (IC15) contains 2077 cropped images with relatively low resolutions and multi-oriented texts. For a fair comparison, we discard the images that contain non-alphanumeric characters, which results in 1811 images.

Figure 4.3 shows some visual examples from *Synth90K* and *SynthText*, where large variations of backgrounds and texts can be observed in the images between two datasets. Therefore, we can conclude that two datasets are of Non-IID. All the training and validation images are scaled to the size of 100*32 in order to fit the model using a mini-batch and accelerate the training process. Testing images in Table 4.1 and 4.3 are scaled proportionally to match the height of 32 pixels. While in Figure 4.4, 4.5 and 4.6, testing images are processed in the same way as what training and validation images do aforementioned. The testing images whose label lengths are less than 3 or more than 25 characters are dropped out due to the limitation of CTC. We deploy *Synth90K* and *SynthText* datasets on two separate clients. On the local training nodes, *AdaDelta* is used for back-propagation optimisation and the initial learning rate is set to 0.05. Regarding HE we set 128 as the key size in bits. The whole gradient is split into 100 pieces and $\hat{p} = 0.9$. Our method is implemented using *Keras* and *Tensorflow*, and the source code is publicly available to ensure reproducibility, which can be run in a distributed multiple GPUs setup.

4.3.2.2 Result and Discussion

Table 4.1 shows the comparison results on testing datasets with different training hyper-parameters including batch size, and the number of epochs. The results of the first row (CRNN*) and the second row (CRNN) are produced by the original CRNN model without using FL framework, where CRNN* corresponds to accuracies reported by its authors in [183] and CRNN corresponds to the results reproduced by us. Compared to the original CRNN model, FedAvg shows a decent amount of improvement. For example, the FedAvg model with a batch size of 800 and epoch of 1 reports 86.67% on *IIIT5k* dataset, where an improvement of 1.19% is achieved compared to CRNN that is of 85.48% using the same setting. An improvement of

4. Protected Gradient Boosting

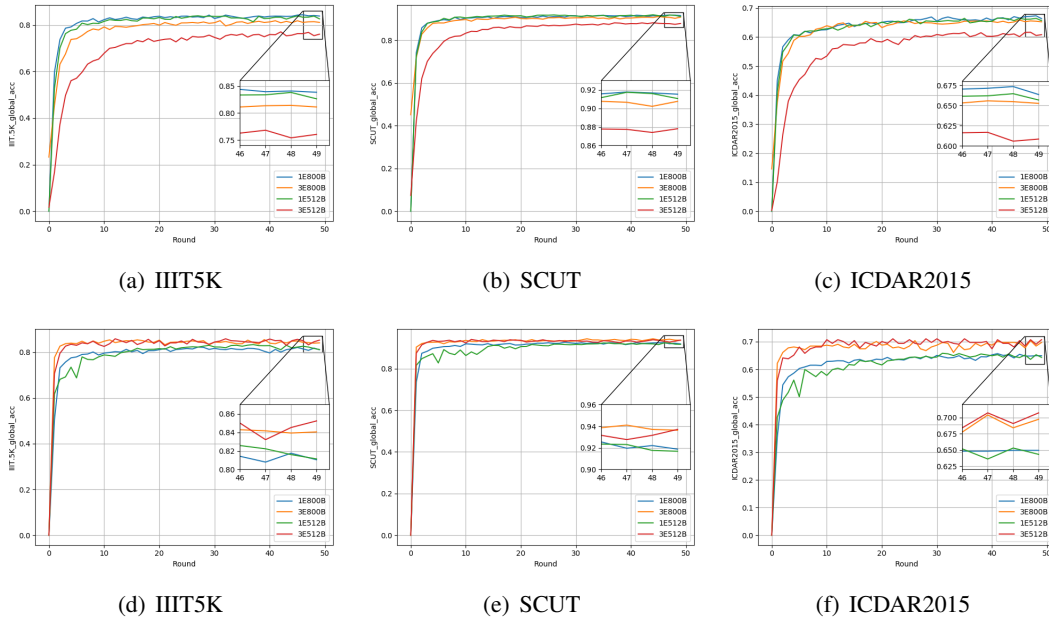


Figure 4.4: Testing accuracy of FedAvg (first row) and FedBoosting (second row) models over rounds with datasets of *IIIT5K*, *SCUT* and *IC15*. “1E800B” means the model is trained on the client with 800 batch size and 1 epoch. All samples in these testing parts are resized to 100*32, which is different from the processing in Table 4.1.

1.65% of FedAvg with the batch size of 512 and epoch of 1 can be observed on *IC15* dataset. More importantly, the proposed FedBoosting achieves the highest accuracy across all the four testing datasets, where 87.62%, 91.60%, 94.65% and 75.99% are reported on *IIIT5k*, *SVT*, *SCUT* and *IC15* respectively. In addition, our method outperforms both FedAvg and non-FL methods by significant margins. More qualitative results are shown in Table 4.2.

It can be observed that the FedAvg models with the bigger batch size and smaller epoch have better performance. In other words, the models perform better when model integration occurs more frequently, which however will increase the communication cost. In Table 4.1, the model with 256 batch size and 1 epoch even produces no result due to model divergence after a few rounds of integration. The potential reason could be the extreme differences in parameters that are learned on each local machine. Figure 4.4 shows the comparison of convergence curves between FedAvg and proposed FedBoosting. The convergence curves of FedAvg model with smaller batch size or larger epoch are always lower than the curves of larger batch size or smaller number of epochs. For example, by the strategy of FedAvg, the model with a batch size of 800 and epoch of 1 (iterated 6,689 and 9,030 times per epoch on *SynthText* and *Synth90K* datasets

respectively) performs significantly better than the model with a batch size of 512 and epoch of 3 (iterated 10,452 and 14,110 times per epoch on *SynthText* and *Synth90K* datasets respectively) on *IIIT5K* testing dataset. On the other hand, the accuracy curves of FedBoosting models (see Figure 4.4 second row), do not have such an issue. Therefore, we can conclude that the boosting strategy we propose can overcome the model collapse issue of FedAvg to a great extent.

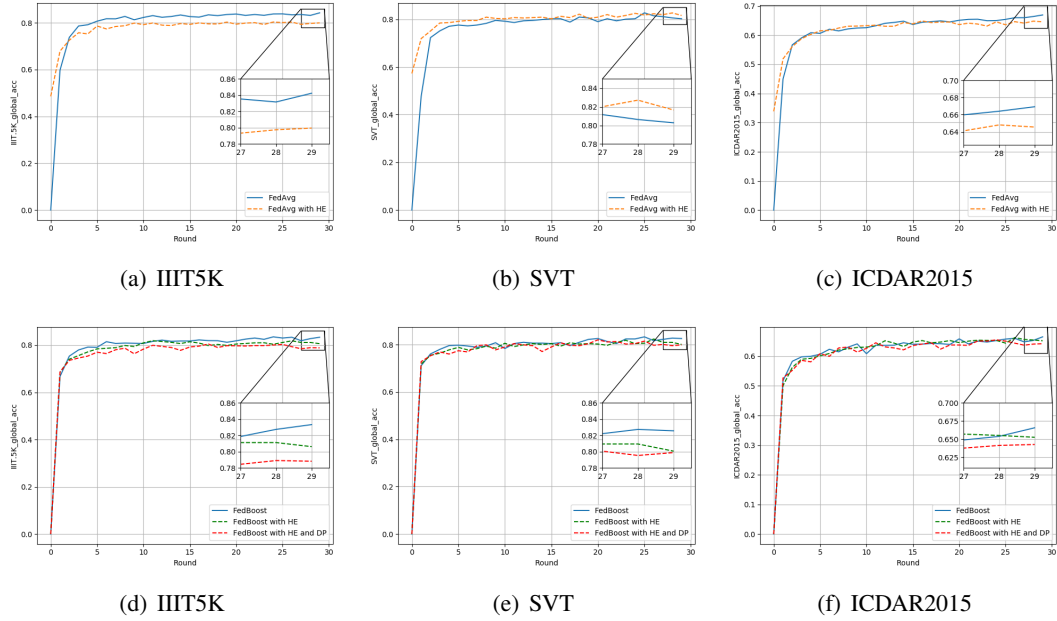


Figure 4.5: Testing accuracy of FedAvg (first row) and FedBoosting (second row) models with and without using encryption protocol over training rounds on *IIIT5K*, *SVT* and *IC15* datasets.

Table 4.3: Recognition accuracies (%) on three testing datasets. All experiments are using a batch size of 800 and epoch of 1.

Method	Encryption	IIIT5K	SVT	IC15
FedAvg	N/A	86.67	89.60	72.37
	HE	86.40	88.00	73.00
FedBoosting	N/A	87.62	89.20	75.99
	HE	85.12	88.40	72.70
	HE+DP	85.00	88.80	72.37

Table 4.3 provides the comparison results on three testing datasets (*IIIT5K*, *SVT* and *IC15*) with different FL gradients merging methods and encryption modes under the hyper-parameters of 800 batch size and 1 epoch. The results of FedAvg illustrate that by using HE, although it has slight precision loss in the processing of dividing the whole gradient into many pieces, accuracy

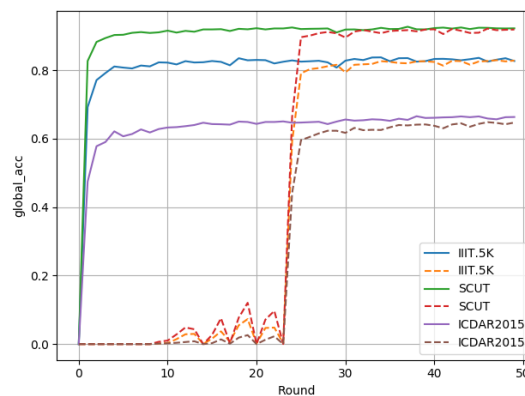
4. Protected Gradient Boosting

has nearly no reduction on testing datasets. Even it has accuracy raising on *IC15* dataset from 72.37% to 73.00%. On the other two testing datasets, the losses of accuracy are 0.27% and 1.6% separately. Same for FedBoosting models, testing results show a slight accuracy reduction which can be tolerant when only using HE. While adding DP into FedBoosting with HE, it has nearly no affection on accuracy. As DP encryption is only used to encrypt local gradients between clients for evaluation and get the results on all clients' validation datasets, DP has little impact on global gradients generating. However, testing results have a fall down between common FedBoosting and encrypted FedBoosting models, *e.g.* accuracy reduced from 87.62% to about 85% on *IIIT5K* and also have an approximately 3% on *IC15*. We think that is normal fluctuation for training DL models. Although all three testing accuracies have different degrees of reduction, from the curve lines in Figure 4.5, the accuracy climbing trend presents the differential under different encryption modes. It can be observed that differentials on most testing datasets are rather small. Please note that all samples in Figure 4.5's testing parts are resized to 100*32, which is different from the processing in Table 4.3 where the samples are scaled proportionally to match with the height of 32 pixels.

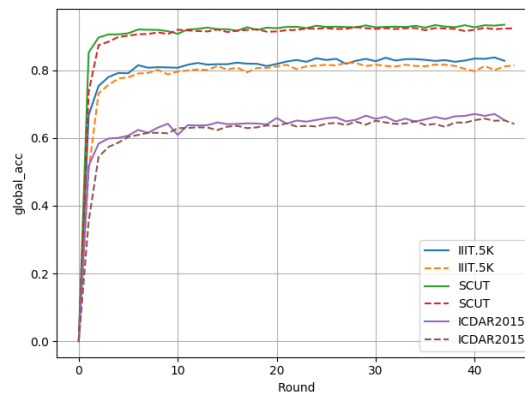
4.3.2.3 Performance Comparison

We further consider that the reason for the divergence issue is the quality of datasets, see Figure 4.3. That is to say, each local model trained on different private datasets has surely different generalisation abilities. In our experiments, aggregating the global model crudely by averaging all the weights of local models may cause a decrease in generalisation ability, especially when the local updating iteration number is large (*i.e.* small batch size or large epoch number). So the proposed FedBoosting prefers to give a more fair weight instead of a mean value by trading off the training and validation performance of a local model. Following this thought FedBoosting first considers each model's validation results on every client's validation dataset, then collaborates with training results to compute the weights of local models. The reason we think over training results is that usually, a local model trained on a high-quality dataset has a nice fitness, while it may perform badly on poor-quality validation datasets. It is unfair to say this model has a poor generalisation ability only considering its validation performance on different quality datasets. Inversely, a model that is trained on a poor-quality dataset may perform very well on a high-quality validation dataset as well, but we do not want this kind of local model to occupy too much of the global model. To this end, to leverage the performance of a local model, we first sum the validation results as a reference representing

the local model’s generalisation ability. Furthermore, training results are taken into account to rectify the reference to obtain the final weights for each local model. It is observed in our experiments that the weights are about 55% for the local model trained on *Synth90K* dataset and 45% on *SynthText* dataset, which is reasonable cause we can see the accuracy results in Table 4.1 that CRNN models trained on *Synth90K* dataset always obtain better performance comparing with those trained on *SynthText* dataset. While if we get rid of training results, the weight for the local model trained on *Synth90K* dataset would be smaller than which on *SynthText* dataset.



(a) 256 batch size and 1 epoch



(b) 800 batch size and 1 epoch

Figure 4.6: Performance comparison on how the training results affect the model performance. Solid lines refer to global models aggregated with training and validation results, while dash lines refer to global models aggregated only with validation results.

To prove the above idea in FedBoosting, a performance comparison is given here. It is commonly accepted that generalisation ability is a good metric for judging a model’s perfor-

mance, whereas only considering generalisation ability is not feasible for our proposed method FedBoosting. Otherwise, it is impossible as well to deploy our method only considering training results and get rid of validation results, which may lead to an extremely unfair situation in that the local model trained on *Synth90K* may take a weight up to about 80% for the global model. So the following content is mainly talking about how training results work in FedBoosting. We trained a global model with 256 batch size and 1 epoch under the strategy of FedBoosting without considering training results. As described above, the reason for thinking over training results is to rectify the weight for the local model. From Figure 4.6 (a), we can see that the global model without taking training results gains a delay convergence at round 24. While in other experiments, models all converge quickly and properly under the supervision of training results. In the meantime, the performance of the global model with training results is always better than that without training results. As a supplement, we visualise the global testing accuracy of two models with 800 batch size and 1 epoch in Figure 4.6 (b), one uses training results and the other one does not. Two models converge normally in this case, but the model performance of using training results outperforms all the time. From all the above, we consider that using training results to supervise the local weight is essential in our scenario. To clarify, all testing images during training are resized to 100×32 , which is different to individual testing experiments where testing images are resized to $W \times 32$, where W is proportionally scaled with heights, but at least 100 pixels. That is why accuracies in Figure 4.6 are lower than those in Table 4.1. Please refer to our codes for more details.

4.4 Summary

In this chapter, we proposed FedBoosting a boosting scheme for FL framework to overcome the limitation of FedAvg on Non-IID dataset. To protect against gradient leakage attack, a gradient sharing protocol was introduced using HE and DP. A comprehensive comparison study has been carried out using a synthetic dataset and public text recognition benchmark, which shows superior performance over the traditional FedAvg scheme on the Non-IID dataset. In addition to this, we acknowledge the limitations inherent in our FedBoosting, particularly in relation to the dual-layer encryption component. This limitation arises from the substantial assumption necessitated by the protocol, which presupposes that no client will engage in a conspiracy with the server to disclose the private key. Such an assumption, while theoretically possible, represents a point of vulnerability, as it does not account for the potential threats posed by rogue clients or server. Thus, the reliability of the FedBoosting’s encryption layers fundamentally

depends on this strong assumption, underscoring the need for further investigation into more robust, adversarial-resilient encryption methods within the context of Federated Learning. Our implementation is publicly available to ensure reproducibility, and it can be run in a distributed multiple GPUs setup. Theoretical study on model convergence from multi-parties computing, privacy leakage from gradients, and more efficient quantization methods for gradients are three potential directions worthy of further investigation.

Chapter 5

Leakage Defence with Key-Lock

5.1 Introduction

DNNs marks the start of the fourth paradigm for data-intensive scientific discovery that has achieved numerous world-renowned successes on computer vision, NLP and classic artificial intelligence problems, which greatly improves human's daily life and speeds up industrial innovation. DNNs are data hungry, where increasingly public concerns on data privacy and personal information have been raised in recent years and attracted researchers' great attention. FL, a decentralised framework for collaborative privacy-preserving model training was proposed [91–93, 191], where it consists of a number of training clients and a central server that aggregates locally computed model gradient from individual clients. The sensitive data is only visible to its data owner. However, recent research works have demonstrated that the gradient-sharing scheme is no longer secure in protecting sensitive and private data. Zhu *et al.* [130] proposed DLG, a method to recover the original training data by updating a randomly initialised input image and its associated label to approximate the true gradient. When the training converges, the updated input image is expected to be closer to the true image. DLG is often criticised for being unstable and sensitive to the size of the training batch and image resolution. Zhao *et al.* [138] showed that the ground-truth label can be computed analytically from the gradient of the loss with respect to its *Softmax* layer. Furthermore, Geiping *et al.* [139] proposed IG to improve the stability of DLG using magnitude-invariant cosine similarity measurement as the loss function. It is capable of recovering images up to a resolution of 224×224 and a training batch size of 100 with an acceptable success rate. Our previous work [131] showed the superior performance of GRNN in recovering training data from the globally shared gradient. It consists

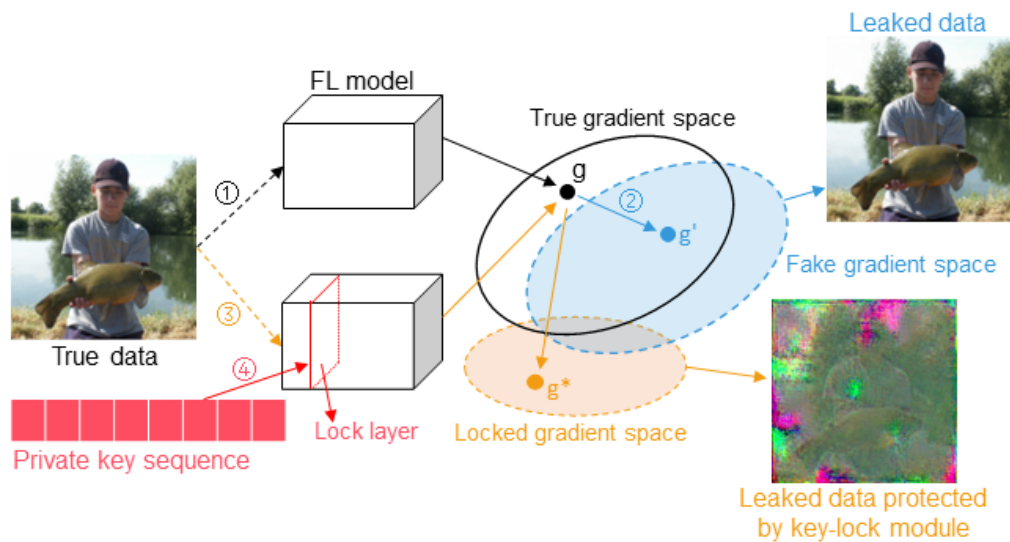


Figure 5.1: Illustration of FedKL: ① Clients feed private training data into the model and generate the true gradient; ② The attacker reconstructs private images from the shared gradient; ③ Clients feed private training data into the model with key-lock module. The attacker can not reveal true images from the locked gradient; ④ A private key sequence is fed into the lock layer for proper inference progression.

of two branches of generative models, one uses a GAN architecture for generating fake training data, and the other one uses a fully-connected architecture for generating corresponding labels. The training data is recovered by a training target of regressing the true and fake gradient that is computed from the generated data pairs. Recently, Li *et al.* [142] proposed GGL that generates similar fake data from a pre-trained GAN. It is derived from the idea of iDLG which is to infer the true label and then update the input sequence of GAN by matching the gradient. Finally, the well-trained input sequence is fed into the GAN model to generate a number of fake images that are similar to the true image. To be noted that GGL only generates similar data given a label, it does not aim to recover the original training data.

The progress of the deep leakage attack leads us to question to what extent can FL system protect data privacy. Recently, there are a number of strategies proposed to protect against such leakage attack, *e.g.* gradient perturbation [130, 131, 192–194], data obfuscation or sanitization [143, 145, 195–198], and mixture methods [115, 144, 199–201]. However, in most cases, trade-offs have to be made between privacy and performance. Particularly, for classic encryption-based approaches, the high computational complexity inherent in cryptographic operators requires an excessive amount of computational resources. Further investigation on the ablation experiment of gradient leakage was carried out by Wei *et al.* [202], where the work quantitatively characterised the correlations of DNN architecture design and federated learning

setting, including batch size, image resolution, activation function and the number of local iterations before gradient exchanging. The findings are directly or indirectly supported by many other works. For example, DLG claims that the activation function must be twice-differentiable; IG can generate images with up to 224×224 resolution; GRNN is capable of recovering the data from a batch size of 256 with a resolution of 256×256 image. It also evaluates the impact of leakage performance by varying the number of local training iterations on the client side.

In this chapter, we first theoretically prove that the feature maps computed from the fully-connected layer, convolutional layer and BN layer contain the private information of input data, where such information also co-exists in the gradient at the backward passing stage. We hypothesise that the gradient leakage attack is possible only when the gradient spaces between the fake model and local model are well aligned. Therefore, we propose *FedKL*, a key-lock module which is capable of differentiating, misaligning and locking the gradient spaces with a private key, meanwhile keeping the federated aggregation the same as the typical FL framework. In summary, we reformulate the scale and shift processes in the normalisation layer. A private key, *i.e.*, randomly generated sequence, is fed into two fully-connected layers, and the outputs are the privately owned coefficients for the scale and shift processes. Both theoretical analysis and experimental results show that the proposed key-lock module is feasible and effective in defending against the gradient leakage attack as the consistency of private information in the gradient is obfuscated so that the malicious attacker cannot formula the forward-backwards propagation without the private key and the gradient of the lock layer. Therefore, it is no longer feasible to reconstruct local training data by approximating the shared gradient in the FL system. Our theoretical and experimental results suggest that the *FedKL* brings the benefits in four aspects:

- Safe - a strong protection strategy against the gradient leakage attack;
- Accurate - negligible degradation of inference performance compared to the model without the key-lock module;
- Efficient - additional computational cost on key-lock module is also negligible;
- Flexible - applicability to arbitrary network architecture.

The rest of this chapter is organised as follows: In Section 5.2, the proposed gradient leakage defence method and private key-lock module are described in detail; Then we will talk about the experimental results in Section 5.3 and discuss the comparison between state-of-the-art attack

methods using our proposed FedKL; The last Section 5.4 reports the conclusion and our future work.

5.2 Proposed Method

In this section, we first prove that the gradient information of a DNN at the back-propagation stage is highly correlated to the input data and its corresponding label in a supervised learning scenario. Then, we theoretically show that by shifting the latent space, the private data cannot be recovered from the gradient without aligning the latent spaces. Therefore, based on these two theoretical findings, the key-lock module is proposed to protect the local gradient from leakage attack. Finally, the novel FL framework with the key-lock module, namely *FedKL*, is presented, which shows how the proposed method protects the collaborative learning system from gradient leakage attack.

5.2.1 Theoretical Analysis on Gradient Leakage

Recently, there have been a number of works showing that the input data information embedded in the gradient enables leakage attack. Geiping *et al.* [139] proved that for any of the fully-connected layers in a neural network, the derivative of loss value with respect to the layer’s output contains the information of input data. By referring to the *Chain Rule*, the input of the fully-connected layer can be computed independently by the gradients of the other layers. In this section, taking typical supervised learning, *i.e.*, image classification, as an example, we first extend this theory to the convolutional layer and BN layer.

Definition 1 Gradient: suppose in a vector or matrix space, a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that maps a vector of length n to a vector with length m : $\mathbf{f}(\mathbf{x}) = y$, $x \in \mathbb{R}^n$; $y \in \mathbb{R}^m$. The gradient is the derivative of a function \mathbf{f} with respect to input \mathbf{x} , and can be presented by the Jacobian Matrix:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \quad (5.1)$$

Definition 2 Chain Rule: concerning the derivative of a function of a function, e.g. $\mathbf{g}(\mathbf{f}(\mathbf{x}))$. With continuity and derivability of the function, the outer function is derivable with respect to the inner function as an independent variable. Then, the inner function is derivable with respect to its independent variable x . According to the Chain Rule, the derivative of \mathbf{g} with respect to \mathbf{x}

is:

$$\frac{\partial \mathbf{g}}{\partial \mathbf{x}} = \frac{\partial \mathbf{g}}{\partial \mathbf{f}} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \quad (5.2)$$

Claim 1 The gradient of the linear neural network is highly related to the input data and ground-truth label.

Proof. Assuming a simple linear regression example, the model is $f(x) : \hat{y} = \theta x$, the loss function is MSE: $\mathcal{L}(x, y) = \frac{1}{N} \sum (\hat{y} - y)^2 = \frac{1}{N} \sum (\theta x - y)^2$, where N is the number of sample, y is the ground-truth. The derivative of loss function \mathcal{L} with respect to the weight θ on a batch of N samples is:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta} &= \frac{\partial}{\partial \theta} \frac{1}{N} \sum_i (x_i \cdot \theta - y_i)^2 \\ &= \frac{1}{N} \sum_i \frac{\partial}{\partial \theta} (x_i \cdot \theta - y_i)^2 \\ &= \frac{1}{N} \sum_i 2 \cdot (x_i \cdot \theta - y_i) \cdot \frac{\partial (x_i \cdot \theta - y_i)}{\partial \theta} \\ &= \frac{1}{N} \sum_i 2 \cdot x_i \cdot (\hat{y} - y_i) \end{aligned} \quad (5.3)$$

In this case, the gradient of the model is positively related to the input to the function and the difference between the predicted label and the ground-truth label. \square

Claim 2 The gradient in a non-linear neural network is highly related to the input data and ground-truth label.

Proof. For a classification task, suppose the model consists of two fully-connected layers, a Rectified Linear Unit (ReLU) activation layer, a *Softmax* layer, and Cross Entropy (CE) loss function. Then the forward process of the model is:

$$\begin{aligned} x &= \text{input} \\ u &= \theta \cdot x + b_1 \\ a &= \text{ReLU}(u) \\ z &= \lambda \cdot a + b_2 \\ \hat{y} &= \text{softmax}(z) \\ \mathcal{L} &= \text{CE}(y, \hat{y}) \end{aligned}$$

5. Leakage Defence with Key-Lock

where $x \in \mathbb{R}^{D_x \times 1}$, $\theta \in \mathbb{R}^{D_o \times D_x}$, $b_1 \in \mathbb{R}^{D_o \times 1}$, $\lambda \in \mathbb{R}^{N_c \times D_o}$, $b_2 \in \mathbb{R}^{N_c \times 1}$, D_x is the dimension of input space, D_o is the number of hidden nodes in the first fully-connected layer, and N_c is the number of class. ReLU is the most commonly used activation function in DL. The derivative of layer ReLU with respect to its input is given by:

$$\frac{\partial \text{ReLU}(x)}{\partial x} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if otherwise} \end{cases} = \text{sgn}(\text{ReLU}(x)) \quad (5.4)$$

The function of *Sigmoid* is $\sigma(x) = \frac{1}{1+e^{-x}}$. And the derivative of it with respect to its input is:

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x)) \quad (5.5)$$

The derivative of loss function \mathcal{L} with respect to the input of *Softmax* z is $(\hat{y} - y)^T$. Then the gradient of the output layer is:

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial \lambda} = (\hat{y} - y)^T \cdot a \quad (5.6)$$

$$\frac{\partial \mathcal{L}}{\partial b_2} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial b_2} = (\hat{y} - y)^T \quad (5.7)$$

This approves that the gradient of the output layer contains the information on the ground-truth label, which is consistent with the point in paper [138, 141]. In Equ. 5.6, \hat{y} is the predicted probability distribution, and y is the one-hot distribution of the ground-truth label. Thus only the correct label produces a negative result of $(\hat{y} - y)^T$. In addition, a , *a.k.a.* feature map as the output of the activation layer, passes the information of input data to the gradient of the output layer. As the value of θ and b_1 are known in typical FL system, the input data can be calculated based on Equ. 5.6, 5.7 and 5.8:

$$a = \begin{cases} \theta \cdot x + b_1 & \text{if } \theta \cdot x + b_1 > 0 \\ 0 & \text{if otherwise} \end{cases} \quad (5.8)$$

Next, the gradient for the first fully-connected layer:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta} &= \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial a} \frac{\partial a}{\partial u} \frac{\partial u}{\partial \theta} \\ &= (\hat{y} - y)^T \cdot \lambda \circ \text{sgn}(a) \cdot x \\ &= \begin{cases} (\hat{y} - y)^T \cdot \lambda \cdot x & \text{if } a > 0 \\ 0 & \text{if otherwise} \end{cases} \end{aligned} \quad (5.9)$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial b_1} &= \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial a} \frac{\partial a}{\partial u} \frac{\partial u}{\partial b_1} \\
&= (\hat{y} - y)^T \cdot \lambda \circ \text{sgn}(a) \cdot 1 \\
&= \begin{cases} (\hat{y} - y)^T \cdot \lambda & \text{if } a > 0 \\ 0 & \text{if otherwise} \end{cases}
\end{aligned} \tag{5.10}$$

Weight $\lambda = \lambda' - \eta(\hat{y} - y)^T a$, where λ' is the last iteration's weight and η is the learning rate. In ReLU layer, certain information is suppressed (setting to 0), see Equ. 5.9. It is worth noting that the non-zero gradients consist of three parts: 1) the difference between the predicted probability distribution and one-hot ground-truth label; 2) the weights of the next fully-connected layer λ ; and 3) the input of the current layer, input data and feature maps. Given the gradient of each layer, each component in Equ. 5.9 & 5.10 is known, therefore the input data x can be recovered. It can conclude that the gradient in a non-linear neural network is highly correlated to the input data and ground-truth label. According to the *Chain Rule*, the input information is also passed through the whole network. \square

Claim 3 The gradient in the CNN is highly correlated to the input data and ground-truth label.

Proof. Consider a classification CNN consisting of a convolutional layer, a ReLU activation layer, a fully-connected layer, a *Softmax* layer, and CE loss function. The forward process of the model is as follows:

$$\begin{aligned}
x &= \text{input} \\
u &= \theta * x + b_1 \\
a &= \text{ReLU}(u) \\
a' &= \text{flatten}(a) \\
z &= \lambda \cdot a' + b_2 \\
\hat{y} &= \text{softmax}(z) \\
\mathcal{L} &= \text{CE}(y, \hat{y})
\end{aligned}$$

The derivatives of \mathcal{L} with respect to λ and b_2 are the same in Claim 2. So we mainly concentrate on the gradient of the convolutional layer:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial u} \frac{\partial u}{\partial \theta} \tag{5.11}$$

5. Leakage Defence with Key-Lock

$\frac{\partial \mathcal{L}}{\partial u}$ is known in Equ. 5.9 & 5.10. According to the convolutional function, we have:

$$u_{(i,j,d)} = \sum_h \sum_w \sum_c \theta_{(d,h,w,c)} \cdot x_{(i+h,j+w,c)} + b_{1d} \quad (5.12)$$

where i and j are the coordinates of the output feature map, d is the index of the convolutional kernel, h and w are the coordinates of the kernel and c is the index of the input channel. Then, the derivative of one pixel of the output feature map against any one specific parameter is:

$$\begin{aligned} \frac{\partial u_{(i,j,d)}}{\partial \theta_{(d,h',w',c')}} &= \frac{\partial (\sum_h \sum_w \sum_c \theta_{(d,h,w,c)} \cdot x_{(i+h,j+w,c)} + b_{1d})}{\partial \theta_{(d,h',w',c')}} \\ &= \frac{\partial \theta_{(d,h',w',c')} \cdot x_{(i+h',j+w',c')}}{\partial \theta_{(d,h',w',c')}} \\ &= x_{(i+h',j+w',c')} \end{aligned} \quad (5.13)$$

All parameters are involved in the computing process of the final output of the model. Therefore, the Equ. 5.11 can be transformed into the derivative of the loss function with respect to any of the parameters in the convolutional layer (see Equ. 5.14). Note that one convolutional kernel is associated with only one output channel, For example, the first kernel with the input generates the first channel of the output, the second kernel with the input results in the second channel of the output, and so on.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta_{(d,h',w',c')}} &= \frac{\partial \mathcal{L}}{\partial u_{(i,j,d)}} \frac{\partial u_{(i,j,d)}}{\partial \theta_{(d,h',w',c')}} \\ &= \frac{\partial \mathcal{L}}{\partial u_{(i,j,d)}} \cdot x_{(i+h',j+w',c')} \end{aligned} \quad (5.14)$$

Regarding the derivative of the loss function with respect to the bias b_{1d} , we have:

$$\frac{\partial u_{(i,j,d)}}{\partial b_{1d}} = 1 \quad (5.15)$$

$$\frac{\partial \mathcal{L}}{\partial b_{1d}} = \sum_i \sum_j \frac{\partial \mathcal{L}}{\partial u_{(i,j,d)}} \quad (5.16)$$

In the end, as the same as Claim 1 and Claim 2, referring to Equ. 5.6, 5.7, 5.9 & 5.10, the gradient in the CNN is also highly correlated to the input data and ground-truth label. And in our previous work [131], we have demonstrated that higher label inference accuracy always leads to better reconstruction performance of the input data. The idea is also presented in papers [138, 141], whereas the fundamental rationale of such phenomena was not explored. \square

We have proved that the gradients in both convolutional and fully-connected layers contain sufficient information on the input data and their associated ground-truth labels to reconstruct them. The objective of the following is to prove the gradient in BN layer also contains information on input data and the ground-truth label.

Definition 3 BN Layer: For each hidden neuron in a DNN, to avoid the vanishing gradient problem caused by input values falling into the limit saturation zone of the non-linear function, BN layer pulls the values back to a standard normal distribution. So that the input values of the non-linear transformation function fall into a region that is more sensitive to the input.

Suppose there is a mini-batch of input data: $\mathcal{B} = \{u_1 \dots u_N\}$, where N is the batch size. The BN function can be defined as:

$$\mu_{\mathcal{B}} = \frac{1}{N} \sum_n u_n \quad (5.17)$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{N} \sum_n (u_n - \mu_{\mathcal{B}})^2 \quad (5.18)$$

$$\hat{u} = \frac{u - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad (5.19)$$

$$s = \gamma \cdot \hat{u} + \beta \quad (5.20)$$

Claim 4 The gradient in BN layer contains the input data and label information.

Proof. We modify the network in Claim 3 by adding a BN layer after the convolutional layer:

$$x = \text{input}$$

$$u = \theta * x + b_1$$

$$s = \text{BN}_{\gamma, \beta}(u)$$

$$a = \text{ReLU}(s)$$

$$a' = \text{flatten}(a)$$

$$z = \lambda \cdot a' + b_2$$

$$\hat{y} = \text{softmax}(z)$$

$$\mathcal{L} = \text{CE}(y, \hat{y})$$

where γ and β are parameters in BN layer. The derivative of the loss function with respect to γ

and β are:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \gamma} &= \frac{\partial \mathcal{L}}{\partial s} \frac{\partial s}{\partial \gamma} = \frac{\partial \mathcal{L}}{\partial s} \cdot \hat{u} \\ &= (\hat{y} - y) \cdot \lambda \circ \text{sgn}(a) \cdot \hat{u}\end{aligned}\quad (5.21)$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \beta} &= \frac{\partial \mathcal{L}}{\partial s} \frac{\partial s}{\partial \beta} = \frac{\partial \mathcal{L}}{\partial s} \\ &= (\hat{y} - y) \cdot \lambda \circ \text{sgn}(a)\end{aligned}\quad (5.22)$$

Same with Claim 1, 2 and 3, the gradient of BN layer is correlated to the input data and ground-truth label as well. However, based on the *Chain Rule*, to calculate the gradient of the network properly, the derivative of the loss function against the input of the BN layer should be computed. It is non-trivial to calculate $\frac{\partial \mathcal{L}}{\partial u}$ directly. We know that the normalised \hat{u} , variance value $\sigma_{\mathcal{B}}^2$ and mean value $\mu_{\mathcal{B}}$ are all related to the input u . So we first compute the derivative of $\frac{\partial \mathcal{L}}{\partial \hat{u}}$, $\frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2}$ and $\frac{\partial \mathcal{L}}{\partial \mu_{\mathcal{B}}}$:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \hat{u}} &= \frac{\partial \mathcal{L}}{\partial s} \frac{\partial s}{\partial \hat{u}} = \frac{\partial \mathcal{L}}{\partial s} \cdot \gamma \\ &= (\hat{y} - y) \cdot \lambda \circ \text{sgn}(a) \cdot \gamma\end{aligned}\quad (5.23)$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2} &= \frac{\partial \mathcal{L}}{\partial s} \frac{\partial s}{\partial \hat{u}} \frac{\partial \hat{u}}{\partial \sigma_{\mathcal{B}}^2} \\ &= -\frac{1}{2} \frac{\partial \mathcal{L}}{\partial \hat{u}} \cdot (u - \mu_{\mathcal{B}}) \cdot (\sigma_{\mathcal{B}}^2 + \varepsilon)^{-\frac{3}{2}}\end{aligned}\quad (5.24)$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mu_{\mathcal{B}}} &= \frac{\partial \mathcal{L}}{\partial \hat{u}} \frac{\partial \hat{u}}{\partial \mu_{\mathcal{B}}} + \frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2} \frac{\partial \sigma_{\mathcal{B}}^2}{\partial \mu_{\mathcal{B}}} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{u}} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \varepsilon}} + \frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{-2(u - \mu_{\mathcal{B}})}{N}\end{aligned}\quad (5.25)$$

Then we have:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial u} &= \frac{\partial \mathcal{L}}{\partial \hat{u}} \frac{\partial \hat{u}}{\partial u} + \frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2} \frac{\partial \sigma_{\mathcal{B}}^2}{\partial u} + \frac{\partial \mathcal{L}}{\partial \mu_{\mathcal{B}}} \frac{\partial \mu_{\mathcal{B}}}{\partial u} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{u}} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \varepsilon}} + \frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(u - \mu_{\mathcal{B}})}{N} + \frac{\partial \mathcal{L}}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{N}\end{aligned}\quad (5.26)$$

As in the typical FL, each layer’s gradient is known to a malicious server. Therefore, every variable in the above equations can be calculated. According to Equ. 5.14 & 5.21, the gradient contains rich information of input data in not only shallow but also deep layers. On the other hand, in Equ. 5.23 & 5.21, γ is a trainable parameter which is highly related to the input feature map \hat{u} . And this can also be observed in Equ. 5.24 & 5.25 as well. As a result, in Equ. 5.26, for the derivative of the loss function with respect to the output of convolutional layer u , the input data information is sufficiently rich, which enables deep leakage attack. \square

In this section, we discovered that the private input data or feature maps can be inferred from the gradient, using the previous layers’ gradient and the current layer’s weight. This discovery offers the essential understanding of how gradient leakage works by matching with the true gradients. Referring to all the reasoning processes above, we can conclude that for any arbitrary CNN:

Proposition 1 For an image classification task, the gradient of the fully-connected layer, convolutional layer and BN layer contain sufficient private information, so the attacker is capable of reconstructing the input data and ground-truth label by regressing the gradient.

5.2.2 Theoretical Analysis on Leakage Defence

Proposition 2 For a CNN, embedding a key-lock module into the model can prevent the inheritance of the private input information throughout the gradient, where it makes the gradient drop the information of input data.

Proof. We theoretically analysed how the proposed key-lock module prevents the information from inheriting throughout the gradient. Instead of treating λ and β as trainable parameters in BN layer, the key-lock module makes them the output of two fully-connected layers, namely lock, given a private input sequence, namely key. The new trainable parameters are ω , ϕ , and two biases b_2 , b_3 . The network is defined as follows:

$$\begin{aligned} x &= \text{input} \\ k &= \text{key} \\ u &= \theta * x + b_1 \\ \gamma &= \omega \cdot k + b_2 \\ \beta &= \phi \cdot k + b_3 \end{aligned}$$

$$\begin{aligned}
 s &= BN_{\gamma, \beta}(u) \\
 a &= ReLU(s) \\
 a' &= flatten(a) \\
 z &= \lambda \cdot a' + b_4 \\
 \hat{y} &= softmax(z) \\
 \mathcal{L} &= CE(y, \hat{y})
 \end{aligned}$$

The derivative of the loss function with respect to ω and ϕ are:

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial \omega} &= \frac{\partial \mathcal{L}}{\partial s} \frac{\partial s}{\partial \omega} = \frac{\partial \mathcal{L}}{\partial s} \cdot k \cdot \hat{u} \\
 &= (\hat{y} - y) \cdot \lambda \circ sgn(a) \cdot k \cdot \hat{u}
 \end{aligned} \tag{5.27}$$

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial \phi} &= \frac{\partial \mathcal{L}}{\partial s} \frac{\partial s}{\partial \phi} = \frac{\partial \mathcal{L}}{\partial s} \cdot k \\
 &= (\hat{y} - y) \cdot \lambda \circ sgn(a) \cdot k
 \end{aligned} \tag{5.28}$$

The gradient in the lock layer contains the information on the key sequence and feature map. Given the key, the gradient of the lock layer can be inferred, where the input data can be leaked. Therefore, we proposed to retain both the key sequence and the weight of the lock layer privately in our proposed *FedKL*. Once embedding the key-lock module, then Equ. 5.23 can be rewritten:

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial \hat{u}} &= \frac{\partial \mathcal{L}}{\partial s} \frac{\partial s}{\partial \hat{u}} \\
 &= \frac{\partial \mathcal{L}}{\partial s} \cdot (\omega \cdot k + b_2) \\
 &= (\hat{y} - y) \cdot \lambda \circ sgn(a) \cdot (\omega \cdot k + b_2)
 \end{aligned} \tag{5.29}$$

The key sequence k , lock layer's weight ω , and bias b_2 are all confidential. Unlike the case in Section 5.2.1 Claim 4, the components in Equ. 5.24, 5.25 & 5.26 can not be calculated due to unknown γ and β . Similarly, in Equ. 5.21 \hat{u} can be calculated, however, s can not be calculated in Equ. 5.20. Therefore, it is incapable of inferring the input information from the feature map s . The proposed key-lock module prevents the feature map from inheriting the input information and carrying it throughout the gradient when the private key and parameter of the lock module are unknown to the attacker. \square

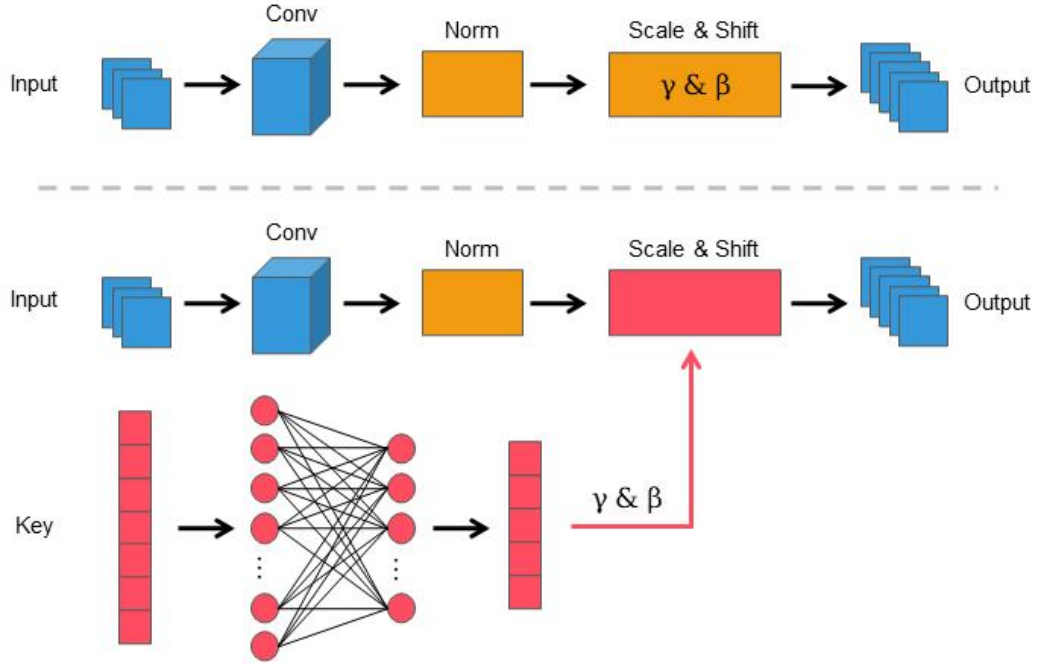


Figure 5.2: The top is the illustration of the general convolution-normalisation block, while the bottom indicates a key-lock module embedded in the block for generating the parameters, γ and β .

5.2.3 Key-Lock Module

We start with a review of the general BN layer. It usually follows a convolutional layer and consists of two processing steps: normalisation, then scale and shift. The coefficients for scale and shift are two trainable parameters updated by back-propagation. Our goal is to prevent the input data information from being passed between gradients. Inspired by the work of DeepIP [203], we propose a key-lock module for FL, namely FedKL, that can be embedded after the convolution-normalisation block, where the scale factor $\gamma \in \mathbb{R}^O$ and the shift bias $\beta \in \mathbb{R}^O$ are no longer two trainable parameters, but the outputs of the key-lock module. O denotes the number of output channels of the convolutional layer. $X_{conv} \in \mathbb{R}^{B \times C \times W \times H}$ is the input of the convolutional layer, where B is the batch size, C is the number of channels, W and H are the width and height of input image, respectively. $W_{conv} \in \mathbb{R}^{C \times O \times K \times K}$ represents the weight in the convolutional layer, where K is the kernel size. We define $X_{key} \in \mathbb{R}^S$ as the input key with a length of S , and $W_{lock} \in \mathbb{R}^{S \times O}$ as the weight of the key-lock module. b is the bias. Therefore, we can formulate the embedding transformation $\mathcal{F}()$ as:

$$\mathcal{F}(X_{conv}, X_{key}) = \gamma \cdot (X_{conv} * W_{conv}) + \beta$$

$$\gamma = X_{key} \cdot W_{lock-\gamma} + b_{lock-\gamma}$$

$$\beta = X_{key} \cdot W_{lock-\beta} + b_{lock-\beta}$$

where $*$ denotes convolution operation, \cdot is the inner product. Note that, γ and β in the general normalisation layer are two trainable parameters, while in our proposed method, they are the outputs of the lock layer given the private key sequence. Figure 5.2 gives the illustration of the general convolution-normalisation block and our proposed key-lock module.

5.2.4 FL with Key-Lock Module

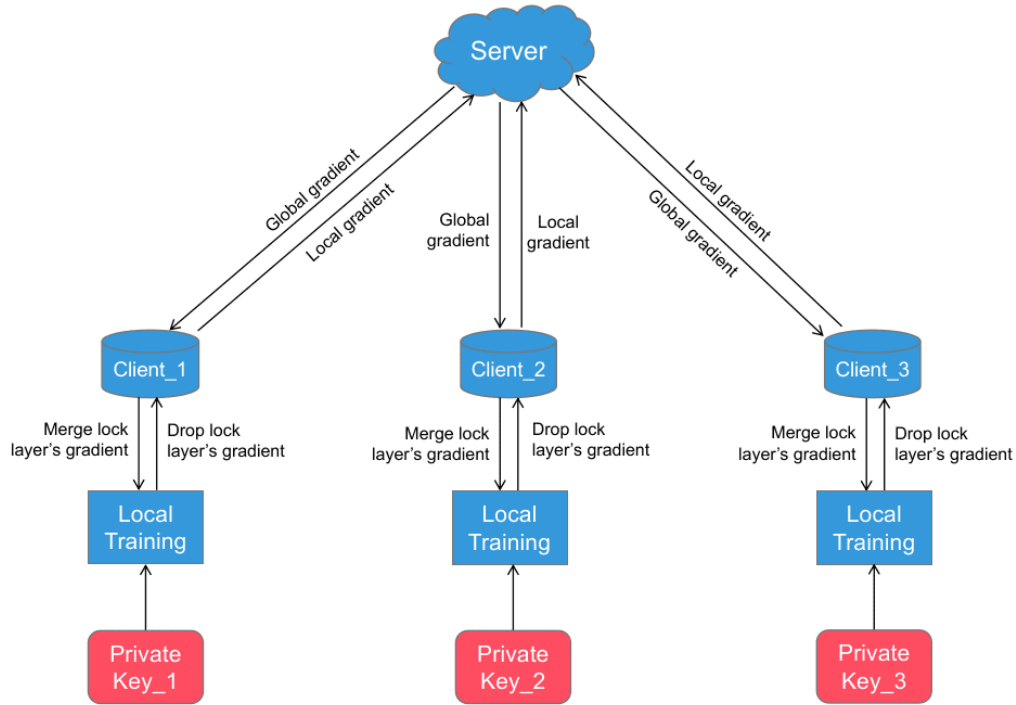


Figure 5.3: FL system with the key-lock module.

In the FL system, the global model is first initialised on the server and then distributed to each client for secured local training. Local gradients from the clients are aggregated to the server to merge into a new global model for the next round of local training. To deploy the key-lock module in the system, the server needs to initialise the model with the key-lock module and then distribute the model to each client. Each client needs to generate its own private key sequence for local training. After local training, the gradient of the model without the lock layer is transferred to the server to generate the new global model. In the whole process, both the

key sequence and lock layer are private and secure locally. The global model does not have information about the key-lock module, so the global model on the server end cannot be used for data reconstruction attack. In the next round of local training, each client will first integrate its own lock layer into the new global model. Figure 5.3 shows the detail of a FL system with the key-lock module. In the next section, we experimentally show that while having the ability of private information protection, the proposed key-lock module has very limited influence on inference performance.

5.3 Experiment and Discussion

In this section, we first introduce the datasets and metrics that are used for benchmark evaluation. Then, we present three sets of experiments to evaluate: 1) the impact of prediction accuracy of DNN with the proposed key-lock module; 2) the efficacy of defending state-of-the-art gradient leakage attacks including DLG, GRNN and GGL; 3) the ablation study of various key hyper-parameters of DNN and training setting.

5.3.1 Benchmarks and Metrics

Four popular public benchmarks used in our experiments are MNIST [158], CIFAR-10 [159], CIFAR-100 [159] and ILSVRC2012 [89]. Four evaluation metrics are used to quantify the quality of the generated image, including MSE, PSNR, Learned Perceptual Image Patch Similarity (LPIPS) [204] and SSIM [205]. As the batch size is set to 1 in order to achieve the highest success rate and the best image quality of gradient leakage attack, the reconstructed image and the true image can be matched directly. MSE measures the pixel-wised L2 difference between the reconstructed image and the true image. Formally, MSE is formulated as $MSE(X, \hat{X}) = \|X - \hat{X}\|_2$, where X indicates the true image and \hat{X} is the reconstructed image. PSNR is an objective standard for image quality evaluation which is defined as the logarithm of the ratio of the squared maximum value of RGB image fluctuation over MSE between two images. The definition is given as: $PSNR(X, \hat{X}) = 10 \cdot \lg(\frac{255^2}{MSE(X, \hat{X})})$. The higher the PSNR score, the higher the similarity between the two images. LPIPS essentially computes the similarity between two image patches by means of some pre-defined networks. We used two neural networks, *VGGNet* [37] and *AlexNet* [36] to perceptually evaluate the similarity of two images. This method is approved to be compatible with human perception. A lower LPIPS score indicates the two images are perceptually more similar. SSIM senses the proximity

of image distortion by detecting whether structural information has been changed. A higher SSIM score implies a better match. The proposed key-lock module is only embedded in the first convolution-normalisation block across all experiments in this chapter. The length of the randomly generated key sequence is 1024. The framework we used to implement the neural network models is PyTorch [163]. Our implementation of proposed *FedKL* is publicly available ¹.

5.3.2 Accuracy Impact

Table 5.1: Testing accuracy(%) from models trained at different settings. The red ones are the highest accuracy and the blue ones are the next highest. C-10 represents CIFAR-10 and C-100 is CIFAR-100. KL means key-lock module. The accuracy results in the Reference row are taken from the relevant papers. “Cen.” is the abbreviation of Centralised

Model	<i>LeNet</i> (32*32)	<i>ResNet-20</i> (32*32)		<i>ResNet-32</i> (32*32)		<i>ResNet-18</i> (224*224)		<i>ResNet-34</i> (224*224)		<i>VGG-16</i> (224*224)		
Dataset	MNIST	C-10	C-100	C-10	C-100	C-10	C-100	C-10	C-100	C-10	C-100	
Cen.	w/o KL	98.09	91.63	67.59	92.34	70.35	91.62	72.15	92.20	73.21	89.13	63.23
	w/ KL	98.07	90.58	67.49	91.05	69.89	93.12	75.90	94.68	78.22	93.84	74.86
FL	FedAvg	98.14	91.20	58.58	91.37	61.91	89.50	68.59	89.27	68.30	88.13	61.91
	FedKL	97.45	88.45	61.97	89.29	64.17	91.66	74.19	93.27	76.61	93.54	73.06
Reference	99.05 [162]	91.25 [39]	-	92.49 [39]	-	-	-	-	-	-	-	-

To evaluate the performance impact, we established two strategies: a) centralised training with/without key-lock module; b) collaborative training in FedAvg system with/without key-lock module. *LeNet* [162] *VGGNet* [37] and *ResNet* [39] are used as the backbone networks for training image classifiers. In Table 5.1, we carried out eleven different settings for the two training strategies and trained forty-four models in total. In general, the best results can be found when the centralised training is used, such as *LeNet* on MNIST has the best performance of 99.05% accuracy from paper [162], *ResNet-20* achieves the highest accuracy of 91.63% on CIFAR-10 and 67.59% on CIFAR-100. By adding the key-lock module, there is no significant drop in terms of model inference performance. In some cases, models with the key-lock module perform even better than the ones without the key-lock module. For example, *ResNet-20* on CIFAR-100 obtains a testing accuracy of 61.97% with FedKL framework, which is 5.79% higher than classic FedAvg model that achieves 58.58%. The highest performance boost can be observed at *VGG-16* on CIFAR-100, where the model from *FedKL* gains an accuracy of

¹<https://github.com/Rand2AI/FedKL>

73.06%, which is 18.43% higher than 61.69% achieved using FedAvg. On the other hand, for the experiments using images with a resolution of 32×32 , the best models are all from centralised settings without the key-lock module. When up-scaling the resolution to 224×224 , the best models change to the centralised setup with our proposed key-lock module. Besides, the results from *ResNet-18*, *ResNet-34*, and *VGG-16* show that models with the key-lock module perform better than those without the key-lock module.

In Table 5.2, we visualised testing accuracies and losses over epochs using different networks trained on CIFAR-10 and CIFAR-100 datasets. The models are all trained in the centralised mode. The performance gap between the models with and without the key-lock module is small. In particular, two kinds of models have the same convergence trend for many cases, such as *ResNet-20* and *ResNet-32* trained on CIFAR-100 and *ResNet-18* trained on CIFAR-10. The superiority of the model with the key-lock module usually happens when using high-resolution images. That is because the key-lock module has two additional fully-connected layers, which bring more trainable parameters compared to the original normalisation layer. Higher image resolution has a more complex feature space, so more trainable parameters lead to better convergence to the model.

In Table 5.1, accuracies from FedKL are the averaged values over all local clients. That is because the gradient of the lock layer is not shared from local clients to the parameter server, and each client holds a different private key sequence. To further illustrate how the key-lock module influences model performance, we experimented with evaluating the global model by inputting a randomly generated key sequence and using the initial weights in the lock layer on the server at each training round. The results are shown in Table 5.3. The accuracy of each client is obtained using their own private input key sequence and well-trained lock layer’s parameter. Average accuracy is the best performance of the three clients in a round. The results in the “Random” row are from the models with the highest average accuracy by inputting a randomly generated key sequence and using the initial lock layer’s weights. In Table 5.3, the performance gap is greater for light networks than for heavy networks. We empirically analysed the relationship between the number of parameters in the lock layer and the total number of parameters in the model. And found a positive relationship between the performance gap and the weight proportion. The performance gap becomes smaller as the proportion of parameters in the lock layer becomes smaller. For example, the simplest network *LeNet* gains the highest parameter proportion of 22.43% and the largest performance gap of 87.00% on the model with the best average performance of 97.45%. The parameter proportion of *ResNet-20* is higher

5. Leakage Defence with Key-Lock

Table 5.2: Testing accuracies and losses over epochs on CIFAR10 and CIFAR100 datasets using different networks. The models are all trained in centralised mode. The red solid lines are the results from models with the key-lock module and the blue dashed lines are those from models without the key-lock module.

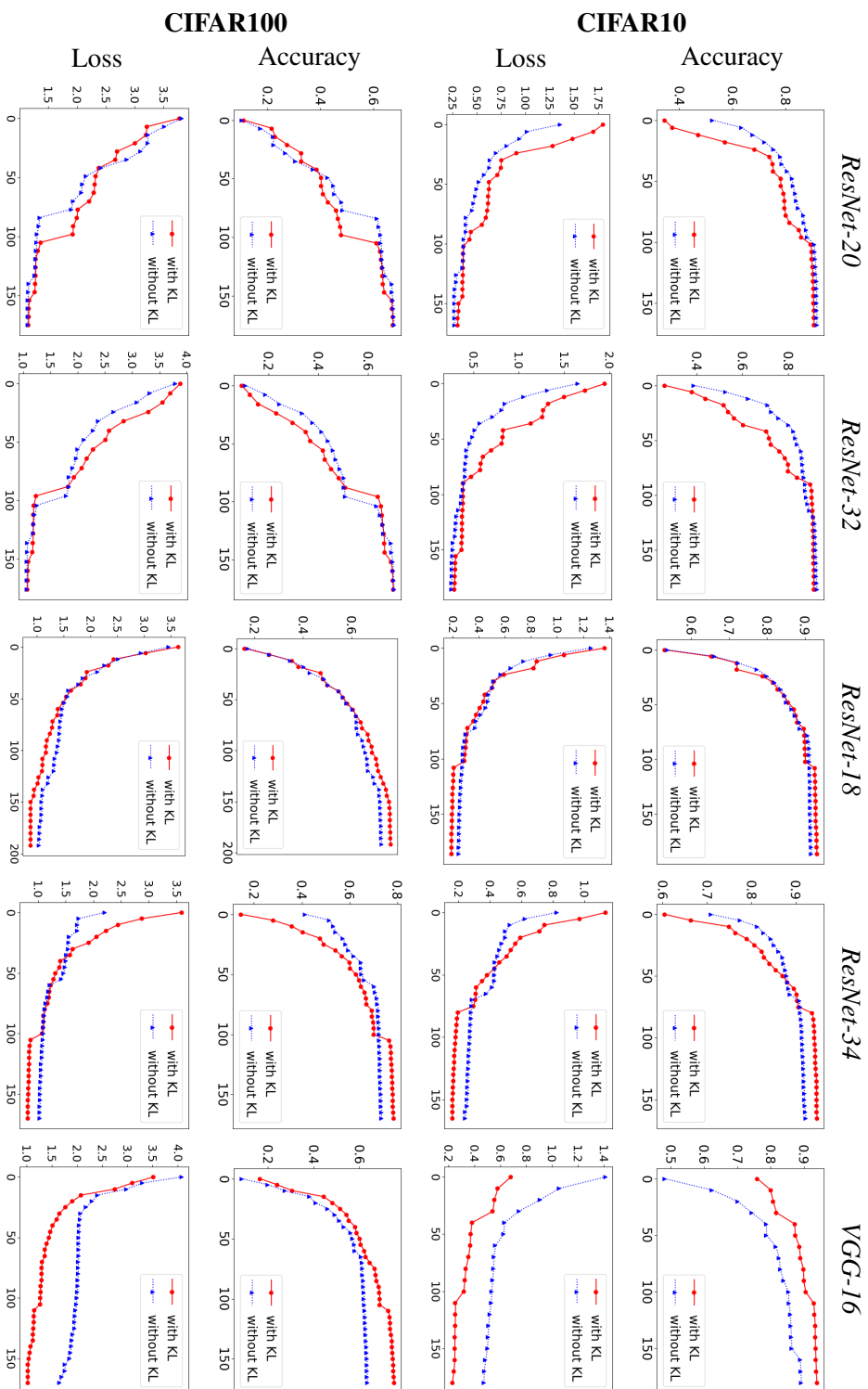


Table 5.3: Accuracy (%) of models trained by FedKL with different input key sequences. The parameter proportion means the ratio of the parameter number in the lock layer over the total parameter number in the model.

Model	Key Source	MNIST		Parameter Proportion
		Accuracy	Gap	
<i>LeNet</i> (32*32)	Client 0	97.88	87.43	22.43%
	Client 1	97.90	87.45	
	Client 2	97.66	87.21	
	Average	97.45	87.00	
	Random	10.45	0	

Model	Key Source	C-10		C-100		Parameter Proportion
		Accuracy	Gap	Accuracy	Gap	
<i>ResNet-20</i> (32*32)	Client 0	88.59	9.42	62.17	33.66	10.49%
	Client 1	87.88	8.71	61.84	33.33	
	Client 2	88.63	9.46	61.99	33.48	
	Average	88.45	9.28	61.97	33.46	
	Random	79.17	0	28.51	0	
<i>ResNet-32</i> (32*32)	Client 0	89.26	13.35	63.92	18.78	6.46%
	Client 1	89.42	13.51	64.28	19.14	
	Client 2	88.83	12.92	64.17	19.03	
	Average	89.29	13.38	64.17	19.03	
	Random	75.91	0	45.14	0	
<i>ResNet-18</i> (224*224)	Client 0	90.23	0.82	74.30	12.70	1.15%
	Client 1	91.78	2.37	74.17	12.57	
	Client 2	89.63	0.22	74.28	12.68	
	Average	91.66	2.25	74.19	12.59	
	Random	89.41	0	61.60	0	
<i>ResNet-34</i> (224*224)	Client 0	93.40	1.63	76.74	6.98	0.61%
	Client 1	92.76	0.99	76.25	6.49	
	Client 2	93.28	1.51	76.57	6.81	
	Average	93.27	1.50	76.61	6.85	
	Random	91.77	0	69.76	0	
<i>VGG-16</i> (224*224)	Client 0	93.58	0.36	72.67	1.07	0.097%
	Client 1	93.40	0.18	73.17	1.57	
	Client 2	93.59	0.37	72.76	1.16	
	Average	93.54	0.32	73.06	1.46	
	Random	93.22	0	71.60	0	

than *ResNet-32*. So *ResNet-20* performs better than *ResNet-32* with a 33.46% accuracy gap against 19.03% on CIFAR-100. However, the performance gap from *ResNet-20* is smaller





































than *ResNet-32* (9.28% VS. 13.38%) on CIFAR-10. The accuracy gap from *ResNet-18* is larger than those from *ResNet-34* (2.25% VS. 1.50% on CIFAR-10 and 12.59% VS. 6.85% on CIFAR-100). *VGG-16* with the lowest parameter proportion of 0.097% has the smallest performance gaps of 0.32% and 1.46% on CIFAR-10 and CIFAR-100, respectively. Apart from the network architecture, the dataset also has a non-negligible effect, *i.e.*, dataset CIFAR-100 with more classes always obtains a larger accuracy gap than CIFAR-10 with fewer classes. This phenomenon occurs in all our experiments.

5.3.3 Defence Performance

To evaluate the defence efficacy against gradient leakage attacks, we re-implemented three state-of-the-art attack methods, DLG [130], GRNN [131] and GGL [142]. The batch size of 1 and resolution of 32×32 for MNIST, CIFAR-10 and CIFAR-100 were used. The resolution of 256×256 was used for ILSVRC2012 instead of 224×224 as GRNN can only generate images with a resolution of the exponential of 2. The original images with lower resolution are upsampled using linear interpolation.

In Table 5.4, selected qualitative results are presented to show the comparative performance of defence efficacy against targeted leakage attacks, including DLG and GRNN with different backbone networks and benchmark datasets. DLG consistently fails using *ResNet-20* and *ResNet-18*, therefore, no result is presented. In the case of using *LeNet*, once the key-lock module is embedded into the network, the reconstructed image is completely perturbed. We found that when BN layer or key-lock module was added into *LeNet*, DLG failed to reconstruct true images in all our experiments since DLG directly regresses the image pixels by approximating the gradient. In other words, DLG can only reconstruct the true image from gradient with explicit input data information (see Section 5.2.1 Claim 1: Equ. 5.3, Claim 2: Equ. 5.9 & Claim 3: Equ. 5.14). But both BN layer and key-lock module re-normalise the feature maps, which leads to misalignment of latent space, furthermore, the ambiguity of input data information in the gradient (See Section 5.2.1 Claim 4: Equ. 5.26 & Proposition 2 Equ. 5.29). According to the quantitative results on gradient leakage attack in Table 5.6, GRNN is more powerful than DLG. However, the proposed key-lock module can still successfully prevent the true image leakage from reconstruction based on the gradient. When employing GRNN on *ResNet-18* with the image resolution of 256×256 , the reconstructed image is also not humanly recognisable for private data identification, although the reconstructed image contains visible but very little information of the true image.

Table 5.4: Comparison of image reconstruction using DLG and GRNN with key-lock module. In this case, the malicious server has no knowledge of the private key sequence and gradient of the lock layer.

Model	Dataset	DLG			GRNN		
		True	w/o KL	w/ KL	True	w/o KL	w/ KL
<i>LeNet</i> (32*32)	MNIST						
	C-10						
	C-100						
<i>ResNet-20</i> (32*32)	MNIST	-	-	-			
	C-10	-	-	-			
	C-100	-	-	-			
<i>ResNet-18</i> (256*256)	C-10	-	-	-			
	C-100	-	-	-			
	ILSVRC	-	-	-			

We found that images generated by GGL are based on two aspects: 1) use the inferred ground-truth label to specify the image classification; 2) finetune the image based on the gradient information to make it as similar as possible to the true image. In Table 5.5, the model with the key-lock module generates samples with the same classification of true images. However, the colour and spatial distribution are entirely different from the true image. That is to say that the key-lock module prevents the GGL from inheriting the semantic and structural information of the true image. For example, in the first row of the table, the samples generated from the GGL model without the key-lock module have similar black grouse with the same green background

5. Leakage Defence with Key-Lock

Table 5.5: Experiments performed on GGL. The backbone network is *ResNet-18* and the dataset is ILSVRC2012 with a resolution of $256 * 256$.



colour and orientation as the true input image. When we embed the key-lock module into the model, GGL can still generate images related to black grouse because the label is constrained by prior knowledge, but the colour distribution and the orientation of the object are very different from the true image. Another one is the sweatshirt example, the model with the key-lock module produced four different colours of sweatshirts. In terms of spatial information, with the exception of the black grouse example, the model with the key-lock module generated three close-up images in the basset hound example, four long-distance images in the pig example, and four images of Bedlington terriers in different poses compared to the images generated by standard GGL. The generated images’ characteristics from the model with the key-lock module are very different from the true image as the latent presentation changes, proving that the information of the true image is adequately protected in the gradient.

Table 5.6: Quantitative comparison of DLG, GRNN and GGL with/without key-lock module. The results are computed from the generated images and their associated true images.

Method	Model	Dataset	MSE↓		PSNR↑		LPIPS-V↓		LPIPS-A↓		SSIM↑	
			w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
DLG	<i>LeNet</i> (32*32)	MNIST	49.97	140.87	32.71	26.64	0.27	0.70	0.13	0.56	0.724	-0.003
		C-10	29.62	98.76	41.14	28.20	0.13	0.48	0.08	0.32	0.724	0.006
		C-100	25.30	101.44	39.85	28.09	0.08	0.49	0.04	0.33	0.859	0.009
GRNN	<i>LeNet</i> (32*32)	MNIST	0.35	103.46	52.78	28.00	0.00	0.69	0.00	0.45	1.000	0.108
		C-10	0.78	92.12	49.29	28.52	0.00	0.49	0.00	0.26	1.000	0.101
		C-100	1.05	100.82	48.65	28.18	0.00	0.49	0.00	0.28	0.999	0.085
	<i>ResNet-20</i> (32*32)	MNIST	1.62	101.28	47.82	28.14	0.04	0.64	0.00	0.36	0.998	0.010
		C-10	9.12	86.42	40.86	28.85	0.00	0.43	0.00	0.24	0.984	0.051
		C-100	20.79	99.25	38.09	28.28	0.00	0.45	0.00	0.25	0.963	0.032
	<i>ResNet-18</i> (256*256)	C-10	15.59	72.55	38.41	29.56	0.01	0.46	0.01	0.54	0.965	0.153
C-100		34.63	85.33	34.13	28.97	0.03	0.48	0.02	0.55	0.917	0.159	
		ILSVRC	13.26	62.87	38.36	30.22	0.04	0.40	0.03	0.48	0.932	0.273
GGL	<i>ResNet-18</i> (256*256)	ILSVRC	63.59	87.22	30.22	28.79	0.40	0.47	0.39	0.46	0.231	0.180

In order to quantitatively compare the performance of state-of-the-art gradient leakage methods with or without the proposed key-lock module, we applied four evaluation metrics to three methods. The metrics we used are MSE, PSNR, LPIPS with *VGGNet*, LPIPS with *AlexNet* and SSIM. In Table 5.6, the results are calculated from the generated images and their associated true images. All methods with the key-lock module were evaluated as worse than those without the key-lock module. For example, DLG with *LeNet* on MNIST has a MSE score of 49.97, while the one with the key-lock module gains 140.87 which is an increase of

5. Leakage Defence with Key-Lock

Table 5.7: Comparison of image reconstruction using DLG, GRNN and GGL with key-lock module. The first case is to share only the private key sequence with the server. Then “Lock” means not sharing the key sequence, but sharing the gradient of the lock layer. “Both” is to share both the key sequence and gradient of the lock layer.

Model	<i>LeNet</i> (32*32)			<i>ResNet-20</i> (32*32)			<i>ResNet-18</i> (256*256)				
Method	GRNN	C-10	C-100	GRNN	C-10	C-100	C-10	GRNN	ILSVRC	ILSVRC	GGL
Shared Info.	Key										
	Lock										
	Both										
True											

181.91%. When embedded key-lock module, GRNN with *LeNet* even gains an increase of 29460% in MSE score on the MNIST dataset. As for the PSNR, most of the models without the key-lock module scored between 35 and 50. However, when the key-lock module is embedded in the model, the PSNR score is approximately 28. In terms of LPIPS and SSIM, the former focuses on the semantic similarity between two images, while the latter focuses on the structural information. The results from both DLG and GRNN perform significantly badly by embedding the key-lock module into the model. Furthermore, unlike DLG and GRNN which generate original true images from gradient directly, the image generated by GGL is similar to the true image in terms of semantics and structure as shown in Table 5.5. We found that GGL with the key-lock module can still generate images with rich semantic and structural information. Nevertheless, the key-lock module reduces the reconstruction capability of GGL by preventing the true image’s information from going through the gradient. So the results from GGL without key-lock module obtain 0.40 and 0.39 on LPIPS using *VGGNet* and *AlexNet*, respectively. When using the key-lock module, the results increase to 0.47 and 0.46, respectively. Although the SSIM score is only 0.231 for normal GGL, it is reduced to 0.180 by embedding the key-lock module into the model. In the end, all the quantitative results in Table 5.6 present that the proposed key-lock module is capable of protecting the private information being leaked in the gradient.

5.3.4 Comparison within Different Sharing Strategies

Our proposed module consists of two components to defend against gradient leakage in the gradient-sharing-based training system, *i.e.*, FL. We theoretically demonstrated that the key-lock module can effectively prevent the inheritance of input data information from embedding into the gradient of the model in Section 5.2.1. In this section, we will experimentally evaluate the influence of the key sequence and lock layer for gradient leakage defence separately. We use three information-sharing strategies for GRNN and GGL with different backbones trained on different datasets. DLG failed to recover any true images across all scenarios, where the possible reason have already been discussed in Section 5.3.3. Therefore, no example from DLG is presented in Table 5.7. We claim that both the key and lock components have the ability to gradient protection, but there exist other reasons that affect the defence performance. For example, network complexity and image resolution. The private key can be used to calculate the gradient of the lock layer, but not vice versa (see Equ. 5.27 & 5.28). Therefore, the gradient in Equ. 5.29 can be inferred theoretically. When the image resolution is fixed at 32×32 , the

GRNN can successfully perform gradient leakage attack by providing a private key only, while it failed to recover any meaningful visual content by providing parameters of key-lock module only. The image quality can further improve by providing both the private key sequence and parameters of the key-lock module. We can observe a similar case when the *ResNet-18* with a resolution of 256×256 is used. We can conclude that even though the key and parameters of the key-lock module both contribute to the gradient protection, the private key information plays a more significant role in defending against gradient leakage attack.

5.4 Summary

In this chapter, we proposed a gradient leakage defence method, named *FedKL*, for the FL system. The proposed key-lock module prevents the input data information from leaking throughout the gradient at the training stage. At first, we theoretically proved the efficacy of the key-lock module in defending against gradient leakage attack. Then, the empirical studies were carried out with three state-of-the-art attack methods. All the quantitative and qualitative comparison results show that by embedding the key-lock module into the model, it is no longer possible to reconstruct input images from the publicly shared gradient. Meanwhile, we discussed the influence of the key-lock module on the model in terms of classification accuracy. The implementation of our proposed *FedKL* is made publicly available to ensure consistent replication and further comparison for researchers in relevant areas.

Chapter 6

Conclusions and Future Work

In this chapter, we first conclude the main contributions we achieved during my Ph.D studies and provide an overview of our work. Then a discussion on the possible extensions of our work is presented.

6.1 Conclusions

In this thesis, we first proposed a state-of-the-art gradient leakage method, GRNN, which utilises GAN to generate fake images and labels, respectively. More specifically, the proposed method is particularly suitable for FL as the local gradient and global model are readily available in the system setting. It consists of two branches for generating fake training data and corresponding labels. It is trained in an end-to-end fashion by approximating the fake gradient that is calculated by the generated data and label to the true gradient given the global model. MSE, WD and TVLoss are used jointly to evaluate the divergence between true and fake gradients. We empirically evaluate the performance of our method on several image classification tasks and comprehensively compared against the state-of-the-art. The experimental results confirm that the proposed method is much more stable and capable of producing images with better quality when a large batch size and resolution are used. To address the gradient leakage problem, we then investigate the gradient aggregation aspect and propose a gradient aggregation protocol in that a DP based linear aggregation method is designed using HE to encrypt the gradients which provides two layers of protection. The proposed encryption protocol only leads to a negligible increase in computational cost. At the same time, instead of treating individual local models equally when the global model is aggregated, we consider the data diversity of local clients

in terms of the status of convergence and the ability of generalisation. Hence, the different client is given a different aggregating percentage instead of averaging the gradients from those clients. At last, we proposed to solve the gradient leakage issue from a new perspective way that by embedding a key-lock module into attribute CNN to break the inheritance of private input information throughout the gradient. Only the locked gradient is transferred to the parameter server for aggregating the global model. The proposed FedKL is robust against gradient leakage attacks.

- We propose a novel data leakage attack method that is capable of recovering private training images up to a resolution of 256×256 and a batch size of 256. The method is particularly suitable for FL as the local gradient and global model are readily available in the system setting. GRNN consists of two branches for generating fake training data and corresponding labels. It is trained in an end-to-end fashion by approximating the fake gradient that is calculated by the generated data and label to the true gradient given the global model. MSE, WD and TVLoss are used jointly to evaluate the divergence between true and fake gradients. We empirically evaluate the performance of our method on several image classification tasks and comprehensively compared against the state-of-the-art. The experimental results confirm that the proposed method is much more stable and capable of producing images with better quality when a large batch size and resolution are used.
- We propose a novel aggregation strategy namely FedBoosting for FL to address the weight divergence and gradient leakage issues. We empirically demonstrate that FedBoosting converges significantly faster than FedAvg while the communication cost is identical to traditional approaches. Especially when the local models are trained with a small batch size and the global model are aggregated after a large number of epochs, our approach can still converge to a reasonable optimum whereas FedAvg often fails in such case. We show the feasibility of our method on two datasets by evaluating the decision boundaries visually. Furthermore, we also demonstrate its superior performance in a visual text recognition task on multiple large-scale Non-IID datasets compared to the centralised approach and FedAvg. The experimental results confirm that our approach outperforms FedAvg in terms of convergence speed and prediction accuracy. It suggests FedBoosting strategy can be integrated with other DL models in the privacy-preserving scenarios.
- We introduce a dual layer protection scheme using HE and DP to encrypt gradients flowing between server and clients, which protect the data privacy from gradient leakage

attack. A DP based linear aggregation method is proposed using HE [2] to encrypt the gradients which provides two layers of protection. The proposed encryption scheme only leads to a negligible increase in computational cost.

- We theoretically prove that the feature maps computed from the fully-connected layer, convolutional layer and BN layer contain the private information of input data, where such information also co-exists in the gradient at the backward passing stage. By referring to the *Chain Rule*, the input of the fully-connected layer can be computed independently by the gradients of the other layers. Taking typical supervised learning, *i.e.*, image classification, as an example, we first extend this theory to the convolutional layer and BN layer.
- We hypothesise that the gradient leakage attack is possible only when the gradient spaces between the global model and local models are well aligned. Therefore, we propose *FedKL*, a key-lock module which is capable of differentiating, misaligning and locking the gradient spaces with a private key meanwhile keeping the federated aggregation the same as the typical FL framework. we reformulate the scale and shift processes in the normalisation layer. A private key, *i.e.*, randomly generated sequence, is fed into two fully-connected layers, and the outputs are the privately owned coefficients for the scale and shift processes. Both theoretical analysis and experimental results show that the proposed key-lock module is feasible and effective in defending against the gradient leakage attack as the consistency of private information in the gradient is obfuscated, so that the malicious attacker cannot formula the forward-backwards propagation without the private key and the gradient of the lock layer. Therefore, it is no longer feasible to reconstruct local training data by approximating the shared gradient in the FL system.

6.2 Future Work

In this thesis, our work is on a line of gradient leakage and gradient leakage defence on FL. We have worked out one way of gradient leakage and two methods of gradient leakage defence. There are three more ideas that we believe can be possible extensions to the work in this thesis:

1. Gradient Leakage to NLP

Same to computer vision, gradient leakage can also take place on NLP when training a DL model under FL. In NLP, Recurrent Neural Network (RNN) is most commonly

used, which is a type of neural network that is well-suited for processing sequential data, such as text. However, RNN suffer from the vanishing and exploding gradient problem as well, which causes the gradient to either disappear or grow excessively large as they are passed through the network. In this thesis, we have given the reasoning process of fully-connected layer, convolutional layer and BN layer. Hence, to prove the relationship between the input private information with the gradient of RNN can be the future work. On the other hand, it is important to carefully design the generative model architecture and choose the appropriate loss function for the gradient leakage to NLP.

2. An Adaptive Gradient Aggregation Method

Although in Chapter 4, we have proposed a method that can adaptively generate gradient aggregation proportion for each client, the communication and computation costings are still expensive. We are thinking of referring Adam [54] to generate the proportion adaptively. Adam is an adaptive algorithm, meaning that it can adjust the learning rates of individual parameters in a model based on historical gradient information. This is in contrast to traditional gradient descent, which has a fixed learning rate for all parameters. Adam works by keeping track of two moving averages of the parameters: the first moment, or the mean, and the second moment, or the variance. It uses these moments to adapt the learning rate of each parameter. The learning rate is adjusted based on the ratio of the first and second moments, with a larger ratio indicating a larger learning rate and vice versa. It requires relatively little tuning of hyperparameters, as it includes default values for the learning rate and other parameters that work well in many cases. It is also computationally efficient and easy to implement.

3. Partial Gradient Leakage

During conducting the experiments for Chapter 5, we found a phenomenon that both DLG and GRNN are capable of reconstructing input images using a partial gradient, which inspired us to explore the latent relationship between the private input information with the gradient. However, we believe that there is still valuable work to be mined along the partial gradient leakage direction. For example, can we infer the gradient of the whole model by giving the structure and partial gradient of the neural network? In our hypothesis, the answer is "YES" because of the experimental results we get that partial gradient leakage is feasible. Inferring the whole gradient from the partial gradient can

help us better understand the latent relationship of gradient within different layers. It can also help us to dig out other reasons for gradient leakage.

6.3 The End

The advent of large language models (e.g. ChatGPT) in the AI Generated Content (AIGC) era has raised several ethical concerns related to monopoly, especially when few giant organisations have the resources and expertise to develop and maintain such models. A primary concern is the concentration of power and control over these AI systems, which could potentially be used to shape public opinion, manipulate information, or stifle competition. This monopolistic situation could lead to a lack of transparency, accountability, and accessibility for the broader public. Moreover, the influence of these organisations on the development of AI regulations and policy could further enhance their dominance in the market, exacerbating existing inequalities and stifling innovation from smaller players. FL offers a promising approach to address some of these ethical concerns by enabling a more decentralised and collaborative model of AI development. With FL, data remains on users' devices, and only updates to the model are shared, thus preserving privacy and potentially reducing the influence of any single organisation. This approach could foster a more diverse ecosystem of AI developers, researchers, and users, which would in turn increase accountability and transparency. By encouraging collaboration and promoting the exchange of ideas, FL can help to create a more equitable AI landscape that benefits a wider range of stakeholders and mitigates the risks associated with large model monopolies.

Bibliography

- [1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Neural Information Processing Systems*, 2014, pp. 2672–2680.
- [2] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Information and Communication Technology Academy of Tamil Nadu*. Springer, 1999, pp. 223–238.
- [3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [4] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.
- [5] D. Wang, X. Wang, and S. Lv, “An overview of end-to-end automatic speech recognition,” *Symmetry*, vol. 11, no. 8, p. 1018, 2019.
- [6] J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong, “Robust automatic speech recognition: a bridge to practical applications,” *Academic Press*, 2015.
- [7] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” *Advances in neural information processing systems*, vol. 28, 2015.
- [8] B. Alshemali and J. Kalita, “Improving the reliability of deep neural networks in nlp: A review,” *Knowledge-Based Systems*, vol. 191, p. 105210, 2020.

- [9] K. Chowdhary, “Natural language processing,” *Fundamentals of artificial intelligence*, pp. 603–649, 2020.
- [10] J. Hirschberg and C. D. Manning, “Advances in natural language processing,” *Science*, vol. 349, no. 6245, pp. 261–266, 2015.
- [11] Y. Goldberg, “A primer on neural network models for natural language processing,” *Journal of Artificial Intelligence Research*, vol. 57, pp. 345–420, 2016.
- [12] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, “Transformers: State-of-the-art natural language processing,” in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45.
- [13] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.-Y. Liu, and W.-Y. Ma, “Dual learning for machine translation,” *Advances in neural information processing systems*, vol. 29, 2016.
- [14] J. Zhang, C. Zong *et al.*, “Deep neural networks in machine translation: An overview.” *IEEE Intelligent Systems*, vol. 30, no. 5, pp. 16–25, 2015.
- [15] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.
- [16] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” *arXiv preprint arXiv:1508.07909*, 2015.
- [17] M. Ott, M. Auli, D. Grangier, and M. Ranzato, “Analyzing uncertainty in neural machine translation,” in *International Conference on Machine Learning*. Proceedings of Machine Learning Research, 2018, pp. 3956–3965.
- [18] H. Kaur and V. Kumari, “Predictive modelling and analytics for diabetes using a machine learning approach,” *Applied computing and informatics*, 2020.
- [19] T. Shaikhina, D. Lowe, S. Daga, D. Briggs, R. Higgins, and N. Khovanova, “Machine learning for predictive modelling based on small data in biomedical engineering,” *IFAC-PapersOnLine*, vol. 48, no. 20, pp. 469–474, 2015.

-
- [20] V. Rodriguez-Galiano, M. Sanchez-Castillo, M. Chica-Olmo, and M. Chica-Rivas, "Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines," *Ore Geology Reviews*, vol. 71, pp. 804–818, 2015.
- [21] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare, "Credit card fraud detection using machine learning techniques: A comparative analysis," in *2017 international conference on computing networking and informatics (ICCNi)*. IEEE, 2017, pp. 1–9.
- [22] J. Perols, "Financial statement fraud detection: An analysis of statistical and machine learning algorithms," *Auditing: A Journal of Practice & Theory*, vol. 30, no. 2, pp. 19–50, 2011.
- [23] A. Mehbodniya, I. Alam, S. Pande, R. Neware, K. P. Rane, M. Shabaz, and M. V. Madhavan, "Financial fraud detection in healthcare using machine learning and deep learning techniques," *Security and Communication Networks*, vol. 2021, 2021.
- [24] S. S. Khanal, P. Prasad, A. Alsadoon, and A. Maag, "A systematic review: machine learning based recommendation systems for e-learning," *Education and Information Technologies*, vol. 25, no. 4, pp. 2635–2664, 2020.
- [25] S. B. Aher and L. Lobo, "Combination of machine learning algorithms for recommendation of courses in e-learning system based on historical data," *Knowledge-Based Systems*, vol. 51, pp. 1–14, 2013.
- [26] S. K. Thangavel, P. D. Bkaratki, and A. Sankar, "Student placement analyzer: A recommendation system using machine learning," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, 2017, pp. 1–5.
- [27] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, "Deep learning for health informatics," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 4–21, 2016.
- [28] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

- [29] I. Castiglioni, L. Rundo, M. Codari, G. Di Leo, C. Salvatore, M. Interlenghi, F. Gallivanone, A. Cozzi, N. C. D'Amico, and F. Sardanelli, "Ai applications to medical images: From machine learning to deep learning," *Physica Medica*, vol. 83, pp. 9–24, 2021.
- [30] A. M. Ozbayoglu, M. U. Gudelek, and O. B. Sezer, "Deep learning for financial applications: A survey," *Applied Soft Computing*, vol. 93, p. 106384, 2020.
- [31] R. Culkin and S. R. Das, "Machine learning in finance: the case of deep learning for option pricing," *Journal of Investment Management*, vol. 15, no. 4, pp. 92–100, 2017.
- [32] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning for finance: deep portfolios," *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, pp. 3–12, 2017.
- [33] H. Nguyen, L.-M. Kieu, T. Wen, and C. Cai, "Deep learning methods in transportation domain: a review," *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 998–1004, 2018.
- [34] S.-H. Fang, Y.-X. Fei, Z. Xu, and Y. Tsao, "Learning transportation modes from smart-phone sensors based on deep neural network," *IEEE Sensors Journal*, vol. 17, no. 18, pp. 6111–6118, 2017.
- [35] A. K. Haghighat, V. Ravichandra-Mouli, P. Chakraborty, Y. Esfandiari, S. Arabi, and A. Sharma, "Applications of deep learning in intelligent transportation systems," *Journal of Big Data Analytics in Transportation*, vol. 2, no. 2, pp. 115–145, 2020.
- [36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.
- [37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [40] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

-
- [41] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, “Residual attention network for image classification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3156–3164.
- [42] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of tricks for image classification with convolutional neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 558–567.
- [43] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, “Convolutional neural networks for large-scale remote-sensing image classification,” *IEEE Transactions on geoscience and remote sensing*, vol. 55, no. 2, pp. 645–657, 2016.
- [44] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.
- [45] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, “Object detection with deep learning: A review,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.
- [46] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, “Relation networks for object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3588–3597.
- [47] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.
- [48] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.
- [49] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [50] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, “Lasot: A high-quality benchmark for large-scale single object tracking,” in *Proceedings*

- of the *IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5374–5383.
- [51] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, “Deepdecision: A mobile deep learning framework for edge video analytics,” in *International Conference on Computer Communications 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1421–1429.
- [52] M. Zolfaghari, K. Singh, and T. Brox, “Eco: Efficient convolutional network for online video understanding,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 695–712.
- [53] C. F. Higham, R. Murray-Smith, M. J. Padgett, and M. P. Edgar, “Deep learning for real-time single-pixel video,” *Scientific reports*, vol. 8, no. 1, pp. 1–9, 2018.
- [54] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [55] I. Goodfellow, “Nips 2016 tutorial: Generative adversarial networks,” *arXiv preprint arXiv:1701.00160*, 2016.
- [56] J. F. Nash Jr, “Equilibrium points in n-person games,” *Proceedings of the national academy of sciences*, vol. 36, no. 1, pp. 48–49, 1950.
- [57] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” *Advances in neural information processing systems*, vol. 29, 2016.
- [58] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” *arXiv preprint arXiv:1809.11096*, 2018.
- [59] F. Zhan, H. Zhu, and S. Lu, “Spatial fusion gan for image synthesis,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3653–3662.
- [60] M. Zhu, P. Pan, W. Chen, and Y. Yang, “Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5802–5810.

-
- [61] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, “pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5799–5809.
- [62] Y. Wang, P. Bilinski, F. Bremond, and A. Dantcheva, “Imaginator: Conditional spatio-temporal gan for video generation,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1160–1169.
- [63] M. Chu, Y. Xie, J. Mayer, L. Leal-Taixé, and N. Thuerey, “Learning temporal coherence via self-supervision for gan-based video generation,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 75–1, 2020.
- [64] Y. Li, M. Min, D. Shen, D. Carlson, and L. Carin, “Video generation from text,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
- [65] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, “Mocogan: Decomposing motion and content for video generation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1526–1535.
- [66] S. Azadi, M. Fisher, V. G. Kim, Z. Wang, E. Shechtman, and T. Darrell, “Multi-content gan for few-shot font style transfer,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7564–7573.
- [67] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, “Neural style transfer: A review,” *IEEE transactions on visualization and computer graphics*, vol. 26, no. 11, pp. 3365–3385, 2019.
- [68] W. Xu, C. Long, R. Wang, and G. Wang, “Drb-gan: A dynamic resblock generative adversarial network for artistic style transfer,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6383–6392.
- [69] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi, “Bagan: Data augmentation with balancing gan,” *arXiv preprint arXiv:1803.09655*, 2018.
- [70] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, “Synthetic data augmentation using gan for improved liver lesion classification,” in *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*. IEEE, 2018, pp. 289–293.

- [71] S.-W. Huang, C.-T. Lin, S.-P. Chen, Y.-Y. Wu, P.-H. Hsu, and S.-H. Lai, “Auggan: Cross domain adaptation with gan-based data augmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 718–731.
- [72] N.-T. Tran, V.-H. Tran, N.-B. Nguyen, T.-K. Nguyen, and N.-M. Cheung, “On data augmentation for gan training,” *IEEE Transactions on Image Processing*, vol. 30, pp. 1882–1897, 2021.
- [73] A. Ramesh, A. S. Rao, S. Moudgalya, and K. Srinivas, “Gan based approach for drug design,” in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2021, pp. 825–828.
- [74] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” in *International conference on machine learning*. PMLR, 2017, pp. 2642–2651.
- [75] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [76] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *Proceedings of Machine Learning Research Conference on International Conference on Machine Learning*, 2017, pp. 214–223.
- [77] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.
- [78] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, “Unrolled generative adversarial networks,” *arXiv preprint arXiv:1611.02163*, 2016.
- [79] L. Mescheder, A. Geiger, and S. Nowozin, “Which training methods for gans do actually converge?” in *International conference on machine learning*. PMLR, 2018, pp. 3481–3490.
- [80] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, “Amortised map inference for image super-resolution,” *arXiv preprint arXiv:1610.04490*, 2016.
- [81] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

-
- [82] M. Liang and X. Hu, “Recurrent convolutional neural network for object recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3367–3375.
- [83] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
- [84] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “Sphereface: Deep hypersphere embedding for face recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.
- [85] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4690–4699.
- [86] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, “A survey of deep learning applications to autonomous vehicle control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712–733, 2020.
- [87] J. Fayyad, M. A. Jaradat, D. Gruyer, and H. Najjaran, “Deep learning sensor fusion for autonomous vehicle perception and localization: A review,” *Sensors*, vol. 20, no. 15, p. 4220, 2020.
- [88] A. Boukerche and X. Ma, “Vision-based autonomous vehicle recognition: A new challenge for deep learning-based systems,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–37, 2021.
- [89] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
- [90] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of Machine Learning Research Conference on Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

- [91] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, 2016.
- [92] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, “Federated optimization: Distributed machine learning for on-device intelligence,” *arXiv preprint arXiv:1610.02527*, 2016.
- [93] B. McMahan and D. Ramage, “Federated learning: Collaborative machine learning without centralized training data,” *Google Research Blog*, vol. 3, 2017.
- [94] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, “Federated learning,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.
- [95] L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.
- [96] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.
- [97] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, “Towards a unified view of parameter-efficient transfer learning,” *arXiv preprint arXiv:2110.04366*, 2021.
- [98] C. Shorten, T. M. Khoshgoftaar, and B. Furht, “Text data augmentation for deep learning,” *Journal of big Data*, vol. 8, no. 1, pp. 1–34, 2021.
- [99] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, “A survey of data augmentation approaches for nlp,” *arXiv preprint arXiv:2105.03075*, 2021.
- [100] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. A. Mann, “Data augmentation can improve robustness,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 935–29 948, 2021.
- [101] D. Gao, Y. Liu, A. Huang, C. Ju, H. Yu, and Q. Yang, “Privacy-preserving heterogeneous federated transfer learning,” in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 2552–2559.

-
- [102] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, “Fedhealth: A federated transfer learning framework for wearable healthcare,” *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020.
- [103] S. Saha and T. Ahmad, “Federated transfer learning: Concept and applications,” *Intelligenza Artificiale*, vol. 15, no. 1, pp. 35–44, 2021.
- [104] S. Yue, J. Ren, J. Xin, D. Zhang, Y. Zhang, and W. Zhuang, “Efficient federated meta-learning over multi-access wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 5, pp. 1556–1570, 2022.
- [105] Z. Charles and J. Konečný, “Convergence and accuracy trade-offs in federated learning and meta-learning,” in *International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research*, 2021, pp. 2575–2583.
- [106] A. Fallah, A. Mokhtari, and A. Ozdaglar, “Personalized federated learning: A meta-learning approach,” *arXiv preprint arXiv:2002.07948*, 2020.
- [107] A. C.-C. Yao, “How to generate and exchange secrets,” in *IEEE Symposium on Foundations of Computer Science*, 1986, pp. 162–167.
- [108] O. Goldreich, “Secure multi-party computation,” *Manuscript. Preliminary version*, vol. 78, 1998.
- [109] A. Choudhury, J. Loftus, E. Orsini, A. Patra, and N. P. Smart, “Between a rock and a hard place: Interpolating between mpc and fhe,” in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2013, pp. 221–240.
- [110] P. Ananth, A. R. Choudhuri, A. Goel, and A. Jain, “Towards efficiency-preserving round compression in mpc,” in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2020, pp. 181–212.
- [111] R. Karl, T. Burchfield, J. Takeshita, and T. Jung, “Non-interactive mpc with trusted hardware secure against residual function attacks,” in *International Conference on Security and Privacy in Communication Systems*. Springer, 2019, pp. 425–439.
- [112] V. Goyal, E. Masserova, B. Parno, and Y. Song, “Blockchains enable non-interactive mpc,” in *Theory of Cryptography Conference*. Springer, 2021, pp. 162–193.

- [113] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, “F: Honest-majority maliciously secure framework for private deep learning,” *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 1, pp. 188–208, 2021.
- [114] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, “Crypten: Secure multi-party computation meets machine learning,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 4961–4973, 2021.
- [115] Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, and X. Zheng, “Privacy-preserving federated learning framework based on chained secure multiparty computing,” *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6178–6186, 2020.
- [116] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, “Towards efficient and privacy-preserving federated deep learning,” in *IEEE International Conference on Communications*, 2019, pp. 1–6.
- [117] C. Xu, J. Ren, D. Zhang, Y. Zhang, Z. Qin, and K. Ren, “Ganobfuscator: Mitigating information leakage under gan via differential privacy,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 2358–2371, 2019.
- [118] J. Zhao, Y. Chen, and W. Zhang, “Differential privacy preservation in deep learning: Challenges, opportunities and solutions,” *IEEE Access*, vol. 7, pp. 48 901–48 911, 2019.
- [119] D. Byrd and A. Polychroniadou, “Differentially private secure multi-party computation for federated learning in financial applications,” *arXiv preprint arXiv:2010.05867*, 2020.
- [120] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, “A hybrid approach to privacy-preserving federated learning,” in *ACM Transactions on Autonomous and Adaptive Systems*, 2019, pp. 1–11.
- [121] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, “Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption,” *arXiv preprint arXiv:1711.10677*, 2017.
- [122] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, “Privacy-preserving deep learning via additively homomorphic encryption,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 1333–1345, 2017.

-
- [123] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, “Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning,” in *2020 Annual Technical Conference*, 2020, pp. 493–506.
- [124] H. Fang and Q. Qian, “Privacy preserving machine learning with homomorphic encryption and federated learning,” *Future Internet*, vol. 13, no. 4, p. 94, 2021.
- [125] N. Fernandes, A. McIver, and C. Morgan, “The laplace mechanism has optimal utility for differential privacy over continuous queries,” in *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2021, pp. 1–12.
- [126] J. Dong, D. Durfee, and R. Rogers, “Optimal differential privacy composition for exponential mechanisms,” in *International Conference on Machine Learning*. Proceedings of Machine Learning Research, 2020, pp. 2597–2606.
- [127] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, “A survey on homomorphic encryption schemes: Theory and implementation,” *ACM Computing Surveys (Csur)*, vol. 51, no. 4, pp. 1–35, 2018.
- [128] Z. H. Mahmood and M. K. Ibrahim, “New fully homomorphic encryption scheme based on multistage partial homomorphic encryption applied in cloud computing,” in *2018 1st Annual International Conference on Information and Sciences (AiCIS)*. IEEE, 2018, pp. 182–186.
- [129] B. Hitaj, G. Ateniese, and F. Perez-Cruz, “Deep models under the gan: information leakage from collaborative deep learning,” in *ACM Conference on Computer and Communications Security*, 2017, pp. 603–618.
- [130] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” *Advances in Neural Information Processing Systems*, vol. 32, pp. 14 774–14 784, 2019.
- [131] H. Ren, J. Deng, and X. Xie, “Grnn: Generative regression neural network—a data leakage attack for federated learning,” *ACM Transactions on Intelligent Systems and Technology*, 2021.
- [132] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *IEEE Symposium on Security and Privacy*, 2017, pp. 3–18.

- [133] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, “Towards demystifying membership inference attacks,” *arXiv preprint arXiv:1807.09173*, 2018.
- [134] S. Truex, L. Liu, M. Gursoy, L. Yu, and W. Wei, “Demystifying membership inference attacks in machine learning as a service,” *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 2073–2089, 2021.
- [135] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, “Label-only membership inference attacks,” in *International Conference on Machine Learning*. Proceedings of Machine Learning Research, 2021, pp. 1964–1974.
- [136] G. Zhang, B. Liu, T. Zhu, M. Ding, and W. Zhou, “Label-only membership inference attacks and defenses in semantic segmentation models,” *IEEE Transactions on Dependable and Secure Computing*, 2022.
- [137] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” in *IEEE Symposium on Security and Privacy*, 2019, pp. 691–706.
- [138] B. Zhao, K. R. Mopuri, and H. Bilen, “idlg: Improved deep leakage from gradients,” *arXiv preprint arXiv:2001.02610*, 2020.
- [139] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, “Inverting gradients—how easy is it to break privacy in federated learning?” *arXiv preprint arXiv:2003.14053*, 2020.
- [140] J. Jeon, K. Lee, S. Oh, J. Ok *et al.*, “Gradient inversion with generative image prior,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [141] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, “See through gradients: Image batch recovery via gradinversion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 337–16 346.
- [142] Z. Li, J. Zhang, L. Liu, and J. Liu, “Auditing privacy defenses in federated learning via generative gradient leakage,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 132–10 142.
- [143] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, “Privacy preserving distributed machine learning with federated learning,” *Computer Communications*, vol. 171, pp. 112–125, 2021.

-
- [144] W. Wei, L. Liu, Y. Wut, G. Su, and A. Iyengar, “Gradient-leakage resilient federated learning,” in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 797–807.
- [145] D. Scheliga, P. Mäder, and M. Seeland, “Precode—a generic model extension to prevent deep gradient leakage,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2022*, pp. 1849–1858.
- [146] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, “Scaling distributed machine learning with the parameter server,” in *Symposium on Operating Systems Design and Implementation, 2014*, pp. 583–598.
- [147] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, “Petuum: A new platform for distributed machine learning on big data,” *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015.
- [148] P. Moritz, R. Nishihara, I. Stoica, and M. I. Jordan, “Sparknet: Training deep networks in spark,” *arXiv preprint arXiv:1511.06051*, 2015.
- [149] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, “Firecaffe: near-linear acceleration of deep neural network training on compute clusters,” in *IEEE Conference on Computer Vision and Pattern Recognition, 2016*, pp. 2592–2600.
- [150] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, “Deep gradient compression: Reducing the communication bandwidth for distributed training,” in *The International Conference on Learning Representations, 2018*.
- [151] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *ACM Conference on Computer and Communications Security, 2015*, pp. 1322–1333.
- [152] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *ACM Conference on Computer and Communications Security, 2015*, pp. 1310–1321.
- [153] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in *Advances in neural information processing systems*, vol. 30, 2017.

- [154] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2528–2535.
- [155] M. D. Zeiler, G. W. Taylor, and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning,” in *IEEE International Conference on Computer Vision*, 2011, pp. 2018–2025.
- [156] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language modeling with gated convolutional networks,” in *Proceedings of Machine Learning Research Conference on International Conference on Machine Learning*, 2017, pp. 933–941.
- [157] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.
- [158] Y. LeCun, “The mnist database of handwritten digits,” <http://yann.lecun.com/exdb/mnist/>, 1998.
- [159] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” in *Citeseer*, 2009.
- [160] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” in *Workshop on faces in 'Real-Life' Images: detection, alignment, and recognition*, 2008.
- [161] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” in *British Machine Vision Association*, 2015.
- [162] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [163] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.
- [164] A. Hore and D. Ziou, “Image quality metrics: Psnr vs. ssim,” in *IEEE International Conference on Pattern Recognition*, 2010, pp. 2366–2369.
- [165] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: a simple and accurate method to fool deep neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

- [166] A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, “Sparsefool: a few pixels make a big difference,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9087–9096.
- [167] A. S. Shamsabadi, R. Sanchez-Matilla, and A. Cavallaro, “Colorfool: Semantic adversarial colorization,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1151–1160.
- [168] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, “Privacy-preserving machine learning as a service.” *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 123–142, 2018.
- [169] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, “A generic framework for privacy preserving deep learning,” *arXiv preprint arXiv:1811.04017*, 2018.
- [170] M. Al-Rubaie and J. M. Chang, “Privacy-preserving machine learning: Threats and solutions,” *IEEE Symposium on Security and Privacy*, vol. 17, no. 2, pp. 49–58, 2019.
- [171] J. Liu and X. Meng, “Survey on privacy-preserving machine learning,” *Journal of Clinical and Diagnostic Research*, vol. 57, no. 2, p. 346, 2020.
- [172] H. C. Tanuwidjaja, R. Choi, S. Baek, and K. Kim, “Privacy-preserving deep learning on machine learning as a service—a comprehensive survey,” *IEEE Access*, vol. 8, pp. 167 425–167 447, 2020.
- [173] N. Koti, M. Pancholi, A. Patra, and A. Suresh, “{SWIFT}: Super-fast and robust privacy-preserving machine learning,” in *Security Symposium*, 2021.
- [174] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, “Advances and open problems in federated learning,” *arXiv preprint arXiv:1912.04977*, 2019.
- [175] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” *arXiv preprint arXiv:1812.06127*, 2018.
- [176] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, “Federated multi-task learning,” in *Neural Information Processing Systems*, 2017, pp. 4424–4434.

- [177] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” *arXiv preprint arXiv:1806.00582*, 2018.
- [178] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of fedavg on non-iid data,” *arXiv preprint arXiv:1907.02189*, 2019.
- [179] C. Xu, Z. Hong, M. Huang, and T. Jiang, “Acceleration of federated learning with alleviated forgetting in local training,” *arXiv preprint arXiv:2203.02645*, 2022.
- [180] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [181] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.
- [182] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
- [183] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.
- [184] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2008.
- [185] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *International Conference on Machine Learning*. ACM, 2006, pp. 369–376.
- [186] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Synthetic data and artificial neural networks for natural scene text recognition,” *arXiv preprint arXiv:1406.2227*, 2014.
- [187] A. Mishra, K. Alahari, and C. Jawahar, “Scene text recognition using higher order language priors,” 2012.

-
- [188] K. Wang, B. Babenko, and S. Belongie, “End-to-end scene text recognition,” in *International Conference on Computer Vision*. IEEE, 2011, pp. 1457–1464.
- [189] S. Zhang, M. Lin, T. Chen, L. Jin, and L. Lin, “Character proposal network for robust text extraction,” in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2016, pp. 2633–2637.
- [190] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu *et al.*, “Icdar 2015 competition on robust reading,” in *International Conference on Document Analysis and Recognition*. IEEE, 2015, pp. 1156–1160.
- [191] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, “Communication-efficient learning of deep networks from decentralized data,” *arXiv preprint arXiv:1602.05629*, 2016.
- [192] X. Yang, Y. Feng, W. Fang, J. Shao, X. Tang, S.-T. Xia, and R. Lu, “An accuracy-lossless perturbation method for defending privacy attacks in federated learning,” *arXiv preprint arXiv:2002.09843*, 2020.
- [193] L. Sun, J. Qian, and X. Chen, “Ldp-fl: Practical private aggregation in federated learning with local differential privacy,” *arXiv preprint arXiv:2007.15789*, 2020.
- [194] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, “Soteria: Provable defense against privacy leakage in federated learning from representation perspective,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9311–9319.
- [195] A. T. Hasan, Q. Jiang, J. Luo, C. Li, and L. Chen, “An effective value swapping method for privacy preserving data publishing,” *Security and Communication Networks*, vol. 9, no. 16, pp. 3219–3228, 2016.
- [196] M. A. P. Chamikara, P. Bertók, D. Liu, S. Camtepe, and I. Khalil, “Efficient data perturbation for privacy preserving and accurate data stream mining,” *Pervasive and Mobile Computing*, vol. 48, pp. 1–19, 2018.
- [197] M. Chamikara, P. Bertok, D. Liu, S. Camtepe, and I. Khalil, “Efficient privacy preservation of big data for accurate data mining,” *Information Sciences*, vol. 527, pp. 420–443, 2020.

- [198] H. Lee, J. Kim, S. Ahn, R. Hussain, S. Cho, and J. Son, “Digestive neural networks: A novel defense strategy against inference attacks in federated learning,” *computers & security*, vol. 109, p. 102378, 2021.
- [199] Z. Bu, J. Dong, Q. Long, and W. J. Su, “Deep learning with gaussian differential privacy,” *Harvard data science review*, vol. 2020, no. 23, 2020.
- [200] H. Ren, J. Deng, X. Xie, X. Ma, and Y. Wang, “Fedboost: Federated learning with gradient protected boosting for text recognition,” *arXiv preprint arXiv:2007.07296*, 2020.
- [201] K. Yadav, B. B. Gupta, K. T. Chui, and K. Psannis, “Differential privacy approach to solve gradient leakage attack in a federated machine learning environment,” in *International Conference on Computational Data and Social Networks*. Springer, 2020, pp. 378–385.
- [202] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, “A framework for evaluating gradient leakage attacks in federated learning,” *arXiv preprint arXiv:2004.10397*, 2020.
- [203] L. Fan, K. W. Ng, C. S. Chan, and Q. Yang, “Deepip: Deep neural network intellectual property protection with passports,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [204] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.
- [205] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.