Available but Invisible: Assessing the Feasibility of Applying Privacy-preserving Technologies in Deep Learning

利用可而不可視:深層学習におけるプライバシー保護技術の適用可能性の評価

February, 2024

Tianying XIE
謝　天瀛

Available but Invisible: Assessing the Feasibility of Applying Privacy-preserving Technologies in Deep Learning

利用可而不可視:深層学習におけるプライバシー保護技術の適用可能性の評価

February, 2024

Waseda University Graduate School of Fundamental Science and Engineering

Department of Computer Science and Communications Engineering, Research on Information Security

Tianying XIE
謝　天瀛

Contents

Chapter 1 Introduction 1

1.1 Background 1

1.2 Research Targets 7

1.3 Contributions 8

1.4 Outline 11

Chapter 2 Contribution 1: Efficient Integer Vector Homomorphic Encryption Using Deep Learning for Ciphertext Training in Neural Network 13

2.1 Introduction 13

2.2 Related Work 15

2.3 Background 17

2.3.1 One-hot Encoding 17

2.3.2 Functions of the HE 18

2.3.3 Deep Machine Learning Concepts and Notations 18

2.4 Methodology 19

2.4.1 Efficient Integer Vector Homomorphic Encryption 19

2.4.2 Constructed Neural Network Model 23

2.5 Experiments 26

2.5.1 Dataset 26

2.5.2 Baseline Model 28

2.5.3 Proposed Model 28

2.6 Performance Analysis 29

2.6.1 Accuracy on the Separate Datasets 29
2.6.2 Accuracy on the Improved Algorithm 29
2.6.3 Accuracy on the Absolute Datasets 33
2.7 Conclusion 35

Chapter 3 Contribution 2: Channel-wise Homomorphic Encryption for Ciphertext Inference in Convolutional Neural Network 37
3.1 Introduction 37
3.2 Background 40
3.2.1 Homomorphic Encryption 40
3.2.2 Privacy-preserving Deep Learning 41
3.2.3 Threat Model 42
3.3 Related Work 42
3.4 Methodology 45
3.4.1 Algorithms of the CHE 45
3.4.2 Batch Normalization with Coefficient Merging 59
3.5 Experiment 64
3.5.1 Datasets and Networks 64
3.5.2 Experimental Setup 65
3.5.3 Result 66
3.6 Discussion 71
3.7 Conclusion 73

Chapter 4 Contribution 3: The Trade-offs of Privacy, Utility, and Efficiency in Differential Privacy-Enabled VQ-VAE for Image Generation 75
4.1 Introduction 75
4.2 Preliminary 77
4.2.1 Differential Privacy 77
4.2.2 Generative Model 78
4.2.3 Vector Quantized-Variational AutoEncoder Model 79
4.2.4 Threat Model 79

4.3 Related Work 80
4.4 Methodology 82
4.4.1 Data Flow 83
4.4.2 Implementation for Training the VQ-VAE with the DP. 83
4.4.3 Description for Metrics 84
4.5 Evaluation 87
4.5.1 Experimental Setup 87
4.5.2 Optical Comparison in Latent Space for Processed Data 90
4.5.3 Implement the VQ-VAE Model with and without the DP in the Vanilla Training Flow 92
4.5.4 Implement the VQ-VAE Model with and without the DP in the Revised Training Flow 94
4.5.5 Results Analysis 103
4.6 Discussion 109
4.7 Conclusion 110

Chapter 5 Discussion 113
5.1 Motivation 113
5.2 Trade-offs 114
5.2.1 Significance 114
5.2.2 Limitation 116
5.3 Future Work 118

Chapter 6 Conclusion 121

Acknowledgement 125

Bibliography 127

List of Research Achievements 143

Chapter 1

Introduction

1.1 Background

Over the past decade, the Internet has sparked a new revolution, that of Artificial Intelligence (AI). AI models built on big data have brought numerous conveniences to people's lives, such as online shopping, navigation, personal identification, targeted advertising, etc. These beneficial applications are all based on information data collected and stored from people's daily lives. Advancements in AI have enhanced the potential to harness and derive benefits from collecting private and sensitive data.

Data gathered from sensors, mobile devices, browsers, and wearable technology feeds into Machine Learning (ML) applications across diverse industries, including finance, e-commerce, social media, healthcare, and recommendation systems. Leading service providers, such as Apple, Google, and Amazon, are rapidly engineering a plethora of data-driven ML solutions [1], especially Deep Learning (DL), concentrated in the thesis. Often categorized as the "data user," these service providers utilize vast amounts of data from individuals, the "data owner," to craft personalized services. The result is sent to the "result owner," the relationship among the three parties is shown in Figure 1.1. It is common knowledge that these services accord data owners tangible commercial and political advantages by facilitating tailored recommendations, health tracking, precision advertising, and insightful predictions.

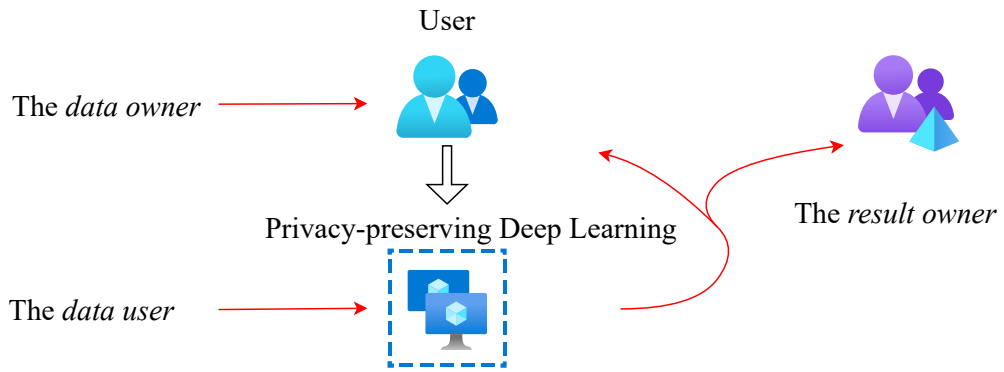


Figure 1.1. The relationship of three parties under the privacy-preserving situation.

Yet, the intrinsic sensitivity of owner data underscores the pivotal concern of privacy. As the synergy between the cloud-based DL and computing crystallizes into robust analytical tools, embedding privacy in the DL models on these platforms becomes indispensable, especially with sensitive datasets in play.

The misuse or unintended use of sensitive and personal data can lead to its exploitation for undue advantages. As we merge vast personal records with the DL algorithms, the outcome becomes unpredictable, casting doubts on the potential insights and the extent of unintentional privacy breaches. Hence, crafting the DL algorithms with a primary focus on privacy is essential. In terms of privacy considerations, three aspects warrant attention [2]:

I The methodology should prevent unauthorized users from leveraging personal data for their gain without the data owner's consent.

II As rational and discerning individuals, everyone possesses private aspects they prefer to remain concealed, irrespective of established legal norms.

III Distinct regions enforce diverse privacy regulations on varying datasets.

Consequently, when subjected to different local laws, the identical dataset can yield contrasting outcomes based on the algorithm employed. The fundamental challenge revolves around achieving an equilibrium between privacy and effectiveness in the DL applications. This aims to maximize the potential of data while ensuring individual privacy remains intact. Given distinct regulatory and applica-

tion needs, compromising on privacy for added utility is not an option. Moreover, privacy safeguards should be systematically implemented rather than relying on arbitrary methods.

Thus, ensuring its privacy becomes paramount since the DL relies significantly on the underlying data when data is sourced from multiple parties in training or from different parties in inference. The model should not reveal any information about the training and inference data, which is named Privacy-preserving Deep Learning (PPDL) [3]. Based on the methods used in papers published over the past decade, this field can be divided into five categories [4].

First of all, Homomorphic Encryption (HE)-based PPDL combines HE [5] with the DL [6] [7] [8] [9] [10] [11] [12]. In the HE-based PPDL, there are typically three stages: training, inference, and result retrieval. During the training stage, a client first encrypts their training dataset using HE and then transfers this encrypted set to a cloud server. The cloud server performs secure training, which yields a trained model, marking the completion of this phase. The client forwards the test dataset to the cloud server in the inference stage. This dataset then serves as an input for the previously trained model, leading to a prediction process that produces an encrypted computational output. After this, the result retrieval phase commences. Here, the cloud server packages and dispatches the encrypted output to the client. The client decrypts this data upon receipt, deriving the final computational result. The threat of the model is the leakage of AI users' privacy to the computing server.

Secondly, Secure Multi-party Computation (SMPC)-Based PPDL introduces a sharing mechanism between the client and the server [13] [14] [15]. In the framework of the SMPC-based PPDL, the process begins with users performing local training on their private data. The next step involves the local training's resulting gradient encoded using secret sharing. This encoded, or "secret-shared," gradient is then distributed to a set of servers. Each server is responsible for accumulating the gradient values received from various users. This aggregation is a critical step in the process. Once aggregation is completed, servers transmit the combined gradient values back to all participating clients. After receiving the result, each client undertakes the task of reconstructing the aggregated gradient from its encoded

form. This reconstructed gradient is then applied to adjust the local training model for the next iteration. This cycle of training, secret sharing, aggregation, and model updating continues iteratively. In the broader context of multi-party computation, secret sharing serves as a foundational technique to preserve data privacy. However, a more robust approach is often adopted in specific contexts of secure two-party computations. Here, a combination of garbled circuits and secret sharing is preferred. This dual approach provides an enhanced level of security compared to using secret sharing alone, making it a more suitable choice for specific two-party computational scenarios where privacy and security are essential. Threats to the SMPC model primarily jeopardize its core function of preserving the confidentiality of an AI model's data. These risks are significant as they can result in the exposure of confidential information, encompassing the data fed into the AI model and the results it generates. This vulnerability extends to all participants engaged in the computation process, potentially compromising the integrity and privacy of the sensitive data involved, especially the leakage of essential parameters in well-trained AI models to the AI users or an adversary disguised as an AI user.

Thirdly, Differential Privacy (DP)-based PPDL puts forward the DP mechanism into the DL with perturbation [16] [17] [18] [19] [20] [21]. As for the structure of the DP-based PPDL, initially, the teacher model is trained using the available training data. This teacher model then facilitates the training of the student model, which in this context is represented as a Generative Adversarial Network (GAN) comprising both a generator and a discriminator. When generating synthetic training data, the generator incorporates random noise. Concurrently, the teacher model educates the student model using public data. The student model orchestrates a zero-sum contest between the generator and the discriminator. Upon its completion, the student model is primed for predictions. When a client forwards a query to the student model, it executes the inference phase and subsequently delivers the prediction results to the client. The primary vulnerabilities associated with the DP model pertain to the potential identification of sensitive training and inference data from the inferential outputs of the AI model. More precisely, there exists a possible risk that users who analyze the model's outputs could deduce confidential information. In response to

this challenge, the deployment of DP is strategically designed to substantially reduce the probability that an individual could ascertain sensitive specifics from the training and inference dataset through meticulous scrutiny of the outputs generated by the AI model.

Fourthly, Secure Enclaves (SE)-based PPDL combines Trusted Execution Environments (TEEs) [22] with the DL [23] [24] [25]. Similarly, a client transmits data to the SE environment. Subsequently, the model provider dispatches the DL model to this enclave. Within the confines of the SE, the prediction is carried out utilizing the client's data and the provided DL model. Post-prediction, the result is relayed to the client. The procedures within the SE are assuredly confidential; any data or models inside remain inaccessible and undisclosed to external entities. The threats of this method mainly focus on hardware dependency, limiting applicability and raising concerns about hardware-level vulnerabilities and the limited computing resources, restricting the complexity of the computation that can be performed securely.

Finally, Hybrid-based PPDL proposes different combinations of the previous methods and other technologies [26] [27] [28]. Due to the various structures resulting from different method combinations, it is necessary to discuss and analyze based on actual situations and make improvements according to the application environment. Therefore, specific model structures will not be described here. The threat of this method is up to the exact combination of multiple methods, so we will not discuss more about it.

In the PPDL model, in other words, the model should not reveal any private information about the training and inference data from the data owner and result owner. The data owner, as previously defined by us, is the party who contributes the data for training; furthermore, the results owner is the party who contributes the data for inference and wants to obtain computational results, such as a result of classification, a prediction of future trends, and so on. Furthermore, the model also should not reveal any sensitive information about itself. The model plays the role of the data user, which accesses and investigates integrated datasets for statistical and research purposes.

There are different approaches above to handling issues. These approaches can be

broadly grouped into two classes [2]: cryptographic-based and perturbation-based approaches. These two classification methods have different emphases. One focuses on specific application methods, while the other concentrates on the macroscopic application mechanisms. Through our research and comparison, we find and believe that the two leading research focuses in the PPDL field currently are methods based on encryption and those based on noise perturbation.

By categorizing each method, we can understand how different the PPDL techniques either rely on cryptographic mechanisms to secure data directly or use perturbation methods to obscure data, ensuring that individual data points are not compromised during the DL process.

The HE enables computations on encrypted data, ensuring that the data remains in a cryptographically secure format throughout the process, which decides the HE-based PPDL should be a cryptographic-based approach. Furthermore, since the SMPC allows multiple parties to jointly compute a function over their datasets without revealing their private information, relying heavily on cryptographic protocols for security, the SMPC-based PPDL should belong to a cryptographic-based approach. Moreover, the SE provides a protected area of memory that isolates the data and code execution from the external environment. The protection mechanisms are generally cryptographic in nature. Thus, the SE-based PPDL is also a cryptographic-based approach.

On the other hand, the DP introduces noise or perturbation to the data or to the algorithm's outputs to prevent the disclosure of individual entries, thereby protecting privacy while maintaining data utility, ensuring that the DP-based PPDL is a perturbation-based approach. As the name suggests, the Hybrid-based PPDL combines elements of both cryptographic methods, like the HE or SMPC, and perturbation methods, like the DP, to provide enhanced privacy protection. The combination depends on the implementation and the targeted privacy and security requirements. Hybrid methods attempt to leverage the strengths of both approaches, often to balance between the robust security of cryptographic methods and the practical utility preservation of perturbation methods.

1.2 Research Targets

This thesis aims to solve the privacy-preserving issue in the DL models that the model should not reveal any private information about the training and inference data from the data owner and result owner. Furthermore, the model also should not reveal any sensitive information about itself. The model plays the role of the data user, which accesses and investigates integrated datasets for statistical and research purposes. In particular, the DL model is utilized chiefly for big data in AI applications, and we focus on the PPDL by cryptographic-based and perturbation-based approaches to minimize the risk of privacy breaches as much as possible and strengthen the awareness of privacy-preserving, thereby achieving "available but invisible." The areas this research targets are described and shown in Figure 1.2.

Cryptographic-based approach. This approach is defined as the cryptography area and countermeasures by encryption and decryption. It can ensure storage security and privacy protection if it is just encryption and decryption naively. However, as mentioned earlier, the AI models need to implement this data to construct models, so they need to process these private and sensitive data directly. The core of this research is how to perform calculations on ciphertext without compromising the data results. Following this comes the HE technology. The HE allows algebraic operations on ciphertext without compromising the results of decryption. That is, it ensures that the results after decryption are almost consistent with those obtained by performing the same algebraic operations on the original data. Cryptographic-based approaches, such as HE, often make significant computational overhead. This can lead to inefficiencies, making these methods impractical for large-scale or real-time applications. Furthermore, integrating cryptographic methods with existing data processing and the DL frameworks can be complex. This complexity arises from modifying standard algorithms to work with encrypted data.

Perturbation-based approach. This approach is recognized by adding noise to smooth out the distribution without changing features in the data. The essence of the DP is how to inject noise, with the main parameters related to the injection

amount being the privacy budget and sensitivity. Sensitivity refers to the degree to which adding or removing a single piece of data within a dataset affects the final result. Meanwhile, the privacy budget refers to the amount of noise added. The smaller the privacy budget value, the more noise is added, thus the higher the level of privacy-preserving. The research problems in perturbation-based approaches, particularly in the PPDL, primarily focus on balancing privacy protection with data utility and system efficiency. Maintaining the utility or accuracy of the data after the noise has been added is the fundamental problem. Furthermore, determining the right amount and type of noise to add is also a complex problem. The challenge is to develop methods for optimal noise addition that adequately balance privacy and utility for the DL models.

1.3 Contributions

The contributions of this thesis are grouped into Contribution 1, Contribution 2 of Target 1, and Contribution 3 of Target 2; the details are as follows.

Contribution 1: Efficient Integer Vector Homomorphic Encryption Using Deep Learning for Ciphertext Training in Neural Network

The DL relies on a large amount of data. In image recognition tasks, such as facial recognition, images contain a large amount of private information. It is essential to ensure that the training and test data are not leaked while obtaining a well-trained model through training. We have researched the Efficient Integer Vector Homomorphic Encryption (EIVHE) scheme and proposed Improved EIVHE (IEIVHE) for better accuracy, as image data contains integer pixels. This method is more suitable for encryption and processing ciphertext on integers compared to other encryption methods. From the EIVHE, the scheme has used a HE algorithm to encrypt the data sets and then train and test the model. The core of the encryption algorithm uses key-switching technology and function polynomialization to achieve the Fully HE (FHE) on the premise of not adding excessive noise, implementing the encryption of non-polynomial functions, and the algebraic operations of ciphertext. The IEIVHE is an improved version of the EIVHE. It demonstrates a specific

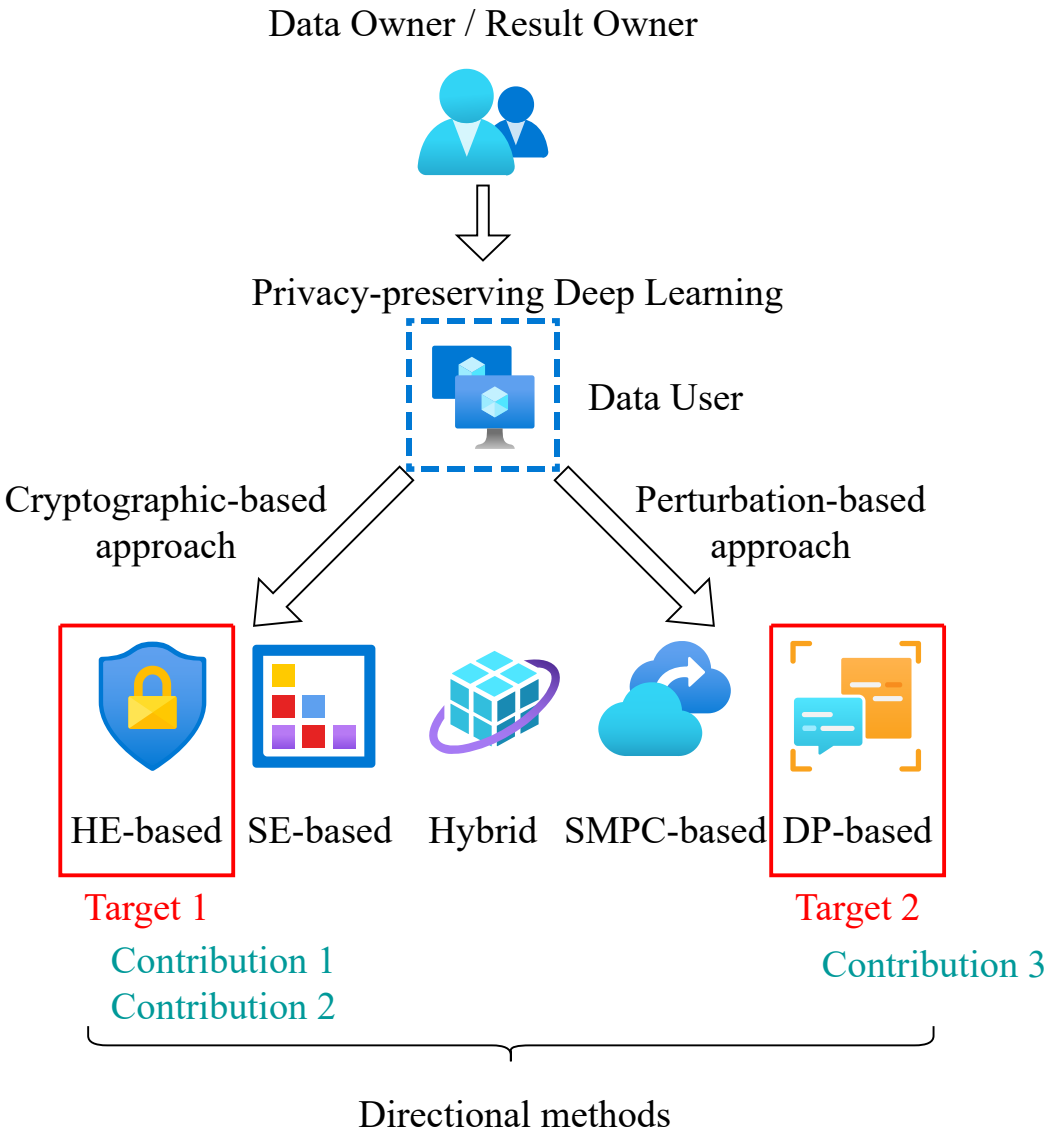


Figure 1.2. The relationship of main research targets and other representatives.

procedure for applying EIVHE to neural networks by combining techniques such as key switching, batch normalization, and matrix transformations and then evaluating its performance in latency and accuracy. The experiments have been tested on a Modified National Institute of Standards and Technology (MNIST) database, and the accuracy is constantly improved through repeated experiments and compared

parameters such as training set accuracy, test set accuracy, time cost and period, etc. In order to make the maximum improvement of accuracy, taking the absolute value of data has been proposed in the IEIVHE, which has improved accuracy 3.08% via the level of data from 86.42% to 89.05% on the MNIST dataset.

Contribution 2: Channel-wise Homomorphic Encryption for Ciphertext Inference in Convolutional Neural Network

This work aims to improve the performance of the image classification of HE-based PPDL by combining two approaches — Channel-wise Homomorphic Encryption (CHE) and Batch Normalization (BN) with coefficient merging. Although these are commonly used schemes, their detailed algorithms and formulations have not been clearly described. The main contribution of the current study is to provide complete and reproducible descriptions of these schemes. By utilizing the CHE, we aim for more efficient data processing at the channel level rather than pixel-by-pixel, reducing inference latency.

More specifically, we propose explicitly tailored algorithms and a computational scheme dubbed "Onion" to facilitate ciphertext inference in Convolutional Neural Network (CNN) using the CHE to address the HE-based ciphertext inference problem in the CNN for higher accuracy and shorter latency. We also present two variations of the convolution, activation, and BN layers in an abbreviation of the layers: Convolution-BN-Activation (CBA) and Convolution-Activation-BN (CAB), which are implementations of the CM, and the two schemes can reduce ciphertext inference latency without increasing the MD. To underscore the stability of these arrangements, we provide mathematical proofs. By merging the BN and activation layer using the CM into a singular mapping layer, we observe a boost in accuracy and a reduction in inference latency. The proposed method achieves the highest accuracy of 99.32% and the shortest latency of 7.76 seconds on the MNIST dataset compared to five previous architectures: the CryptoNets, the light CNN, the HCNN, and so on. And it also attains an accuracy of 76.4% and a latency of 111.91 seconds on the CIFAR-10.

Contribution 3: The Trade-offs of Privacy, Utility, and Efficiency in Differential Privacy-Enabled VQ-VAE for Image Generation

From the perspective of the Perturbation-based approach, in this work, we offer a detailed exploration of a strategy that paves the way for safe data sharing. The synthetic datasets, crafted by our privacy-preserving generative model, can be shared extensively without breaching any privacy norms or ethical boundaries. We showcase an enhanced privacy-preserving Vector Quantized-Variational AutoEncoder (VQ-VAE) framework, improving and speeding up the workflow with a lower privacy budget. For the implementation of the DP, we adopted DP-Stochastic Gradient Descent (DP-SGD) shown in the Tensorflow Privacy library [29] and applied it to our model. This ensures the generation and reconstruction of lifelike images while safeguarding the intrinsic data's privacy and optimizing the privacy budget. Furthermore, we thoroughly examine the balance between privacy, utility, and computational efficacy in our advocated model. Focused on the privacy budget, in the vanilla method, the privacy budgets consumed for four datasets are 3.86, 3.86, 4.94, and 15.53, respectively; in the revised method, the privacy budgets consumed for four datasets are 2.63, 2.63, 2.92, and 10.16, respectively, which reduces the privacy budgets by 31.87%, 31.87%, 40.89%, and 34.58% on the corresponding datasets. The proposed revised training flow can reduce the privacy budget by approximately 34.80% while maintaining a similar generation for images, resulting in a more robust privacy-preserving level.

1.4 Outline

The remainder of this thesis is organized as follows. Chapter 2 details the concept and core technology of implementing the EIVHE and the improved IEIVHE. It also involves applying the algorithm to neural network models for experimental tests, analyzing the resulting performance, discussing improvement plans to enhance accuracy, and verifying its feasibility and effectiveness. Chapter 3 presents the CHE for ciphertext inference in the DL, especially the CNN. The key idea behind the proposed method is shifting pixel-wise encryption to pixel-wise encryption. It is explicitly utilizing several designed algorithms for acceleration and optimization. In Chapter 4, we implement the DP mechanism for a generative model VQ-VAE,

which promotes reducing the privacy budget ϵ of training holding relevant performance. Chapter 5 describes the limitations of privacy-preserving and future work of preserving privacy in the DL. Finally, Chapter 6 summarizes this thesis.

Chapter 2

Contribution 1: Efficient Integer Vector Homomorphic Encryption Using Deep Learning for Ciphertext Training in Neural Network [*1]

2.1 Introduction

Nowadays, Deep Neural Networks (DNNs) have been applied in a significant number of applications, which can be roughly divided into six fields, including image and object recognition, electronic games, voice generation and recognition, imitation of art and style, prediction, and website design modification [30] [31].

However, expanding these fields has revealed limitations in local computing re-

[*1] This chapter is based on the paper: [Efficient Integer Vector Homomorphic Encryption Using Deep Learning for Neural Networks, ICONIP, 2018].

sources. Personal computers are increasingly found inadequate for these demanding tasks, leading many users to turn to cloud computing for enhanced performance. While cloud computing offers significant computational advantages, it also raises critical concerns about data security. A major challenge in cloud-based data processing is ensuring data privacy and usability. Although the performance of computing has been improved, it still faces a problem of how cloud platforms can guarantee the security and privacy of the user's data. Ensuring the privacy of data and its usability is an intractable problem in the process of computing data. Homomorphic Encryption (HE) technology, as a critical method to solve this problem, has been a hot issue in international and domestic academic circles in recent years.

The HE is primarily of two types: Fully Homomorphic Encryption (FHE) and Somewhat Homomorphic Encryption (SWHE). The FHE supports any given function as long as the function can be described by an algorithm and implemented in a computer. The FHE solution is a great solution, but the computational overhead is enormous. Conversely, the SWHE, while only supporting specific functions, presents reduced computational demands and more straightforward implementation. The SWHE scheme is slightly weaker, but it also means that the overhead will be more negligible and more accessible.

Moreover, the four key algorithms – HE on integer vectors [32], SWHE [33], practical FHE [34], and the FHE without bootstrapping [35] – are noted for their efficiency. The DL techniques based on DNNs have achieved significant applications in various domains. There is an excellent risk of disclosing the user's privacy when researchers train a high-performing model with many datasets collected from the users without any protection.

In this chapter, we utilize an Efficient Integer Vector Homomorphic Encryption (EIVHE) [32] scheme using Deep Learning (DL) for the DNN model to address this issue and propose an Improved EIVHE for better accuracy. The EIVHE is a natural extension of the Peikert-Vaikuntanathan-Waters (PVW) [36] method of packing many plaintext elements in a single Regev-type ciphertext [37], which we make the improvement from the original EIVHE based on level of data to the IEIVHE. Image data contains integer pixels, so this method is more suitable for

encryption and processing ciphertext on integers than other encryption methods. We use the EIVHE and IEIVHE to encrypt the dataset, feed the encrypted datasets into a DNN model, and finally obtain the well-trained model for the DNN. It is an innovative bridge between cryptography and deep learning. It aims to protect the user's privacy, combining the State-Of-The-Art (SOTA) DL methods with advanced cryptography [35]

Our contributions are summarized as follows:

- We utilize an EIVHE and propose an improved IEIVHE using the DL to protect users' data privacy in the DNN model.
- To protect privacy in training and test images for our scheme, we exploit the FHE scheme to encrypt the input data using the EIVHE encryption scheme before feeding them into the DNNs.
- We evaluate the EIVHE and IEIVHE on the Modified National Institute of Standards and Technology (MNIST) database, and the experimental results show that we can train the DNN model with encrypted datasets without privacy leakage and achieve an accuracy of 89.05% on the MNIST with the absolute value in the IEIVHE.

2.2 Related Work

The HE was first proposed by Rivest and others [5] in 1978 as a concept: the HE is a method of encryption that allows users to perform certain specific algebraic operations, such as addition, subtraction, multiplication, and division, directly on ciphertext that has been encrypted using the HE. The results of these operations, still in encrypted form, can then be decrypted by the user using the corresponding decryption method to yield plaintext results consistent with those obtained by performing the same algebraic operations directly on the original plaintext. The core idea of the HE is to add random noise to the plaintext, so each time an algebraic operation is performed at the level of the HE, a portion of the noise is added to the ciphertext. Correct decryption results cannot be obtained once the noise exceeds a

set threshold. Hence, researchers have been trying to find ways to reduce the noise.

It wasn't until 2009 that Gentry [38] proposed a fully HE scheme based on ideal lattices, solving a problem that had plagued researchers for years. This method can handle computations of any depth, referred to by the author as "bootstrapping." This technique can reduce the amount of noise in the ciphertext. Still, its computational cost is very high, making it impractical for real-world applications.

Later, to reduce time overhead, researchers proposed a faster HE scheme: Leveled HE (LHE). Without using bootstrapping, this approach reduces computational depth to speed up processing. Furthermore, the DL based on the HE has become an important research method in the field of the DL. To protect user privacy, Shokri and others [3] proposed a Privacy-preserving Deep Learning (PPDL) system based on the HE. This system allows multiple users to train using their local datasets while learning about the DNN model from the joint dataset. The authors primarily made a trade-off between accuracy and gradients, ensuring that as much privacy as possible is protected while sharing user-provided gradients for training.

Similarly, to ensure the feasibility of running the DNNs on encrypted data, Dowlin and other researchers [6] in 2016 introduced a CryptoNets, the first to implement DNN model using the FHE for the PPDL. The authors deployed one of the four most popular HE algorithms, Yet Another Somewhat HE (YASHE) [35], which does not support floating-point operations and instead uses fixed-precision real numbers, converting floating-point numbers into integers at a fixed ratio. Additionally, Ehsan and others [13] focused on how to use the HE for privacy protection in Convolutional Neural Networks (CNNs) in the DL. The authors primarily used approximation methods, utilizing polynomials as activation functions to implement the CNNs. Their CryptoDL model provided efficient, accurate, and scalable privacy-protected predictions.

Furthering research on implementing polynomials as activation functions, Chou and others [14] utilized pruning and quantization schemes to find the optimal polynomial approximation of activation functions, ensuring a more efficient application of the HE in DNN, balancing approximation error with practical usability. On the other hand, based on the framework in CryptoNets [6], Badawi and others [11] ap-

plied Graphics Processing Unit (GPU) acceleration to the FHE technology to realize efficient Homomorphic Convolutional Neural Networks (HCNNs). This means that as many polynomial approximations of activation functions as possible can be used for computation, allowing the use of larger datasets and deeper DNNs.

2.3 Background

2.3.1 One-hot Encoding

In the development of data science projects, datasets commonly feature mixed data types comprising both categorical and numerical columns. Notably, many Machine Learning (ML) models are incompatible with categorical data, necessitating their conversion to a numerical format for model integration, especially the DL. A prevalent issue with categorical data, often represented as string labels, is the potential misinterpretation by the DL models as having an inherent order or hierarchy. To address this, label encoding can be employed, assigning numerical values to these categorical labels, thereby facilitating their effective utilization in the DL algorithms. One-hot encoding is a method employed to convert categorical variables into numerical representations for use in the DL models, which we take to replace the label of structured data. This technique offers several advantages:

- Enabling Numerical Integration: It facilitates the incorporation of categorical variables into models that necessitate numerical inputs, thus broadening the applicability of these models.
- Enhanced Model Performance: By providing a more detailed representation of categorical variables, one-hot encoding can potentially improve model accuracy and efficacy.
- Mitigation of Ordinality Issues: It addresses the challenge of ordinality in categorical data. Ordinality arises when a categorical variable possesses a natural order, such as animals categorized as “cat” and “dog.” One-hot encoding ensures that these inherent orderings do not bias the model’s interpretation of the data.

2.3.2 Functions of the HE

The HE was initially proposed by Rivest, Adleman, and Dertouzous soon after they developed the Rivest–Shamir–Adleman (RSA) public-key cryptosystem [5] [39] [40]. The HE facilitates the execution of operations on plaintexts by applying them to the corresponding ciphertexts, all while keeping the plaintexts confidential [35]. Typically, a HE scheme is composed of four key algorithms: KeyGen, Encrypt, Decrypt, and Evaluate [38] [41]. Based on the established definition of the HE, these four functions can be clearly outlined, as referenced in [42], in the following manner:

- KeyGen. It takes the security parameter w and then produces a secret key s_k and a public key p_k, i.e., $keyGen(w) \rightarrow (p_k, s_k)$
- Encrypt. It takes the public key p_k and a plaintext m as input and produces a ciphertext c of m, i.e., $Encrypt(m, p_k) \rightarrow c$
- Decrypt. It takes the secret key s_k and c as input and produces the plaintext m of c, i.e., $Decrypt(c, s_k) \rightarrow m$
- Evaluate. It takes the public key p_k, a circuit C and a tuple of ciphertext $(c_1, c_2, \cdots, c_n)$ as input and produces the encrypted result c, i.e., $Decrypt(s_k, c) = f(m_1, m_2, \cdots, m_n)$, where f is the functionality that we want to perform.

2.3.3 Deep Machine Learning Concepts and Notations

Given a training dataset, the learning task is to determine these weight variables to minimize a pre-defined cost function such as the cross-entropy or the squared-error cost function [43].

The DL refers to the multi-layer neural network using various DL algorithms to solve the algorithm set's image, text, and other problems. The DL can be classified into the DNNs in broad categories, but implementation has many differences. The core of the DL is feature learning, which aims to obtain hierarchical feature

information through the hierarchical network to solve the critical problems that need artificial design features in the past.

Deep Neural Networks (DNNs), known for their impressive efficacy in various DL tasks, represent parametric functions from inputs to outputs. They compose multiple layers of fundamental elements like affine transformations and basic non-linear functions [44]. By adjusting the parameters of these elements, we can "train" these parametric functions to align with any specific finite collection of input and output examples.

Typically, we divide a DNN into three layers: the input layer, the hidden layer, and the output layer. An input layer is the interface to which we feed our dataset to the network. The hidden layer is the processing unit of the model. The output layer is the result we get from the model. And so-called DL means the network has many hidden layers. Thus, we can regard it as the DL, a process of constant grinding, first defining some standard parameters and then revising them.

2.4 Methodology

2.4.1 Efficient Integer Vector Homomorphic Encryption

We first introduce the used scheme to encrypt datasets, including the encryption scheme and key-switching technique. To encrypt the dataset, the EIVHE algorithm implemented in the DNN model has been clarified in the following outlined in Algorithm 1. The encryption scheme we are examining represents a logical progression from the Peikert-Vaikuntanathan-Waters (PVW) approach, which involves encapsulating multiple plaintext elements within a single Regev-style ciphertext, as detailed in [36] [37].

Encryption Scheme.

At first, the following describes the algorithm of EIVHE developed by Zhou et al. [32]. Let $\mathbf{x} \in \mathbb{Z}_p^n$ be an integer vector to encrypt, where n denotes the length of the vector and p denotes alphabet size. Let $\mathbf{c} \in \mathbb{Z}_q^{n+1}$ be the ciphertext of $\mathbf{x}$ with length $n+1 > n$ and alphabet size $q \gg p$. The secret key is a matrix $\mathbf{S} \in \mathbb{Z}_q^{m \times n}$ and

it satisfies

$$\mathbf{Sc} = w\mathbf{x} + \mathbf{e} \tag{2.1}$$

where noise matrix $\mathbf{e} \in \mathbb{Z}^{m\times n}$. Here, w is an integer parameter such that $w > 2|\mathbf{e}|$. The encryption of $\mathbf{x}$ involves identifying a ciphertext $\mathbf{c}$ that ensures $\mathbf{Sc}$ complies with Equation 2.1. Based on the secret key $\mathbf{S}$, decryption of the ciphertext $\mathbf{c}$ as per Equation 2.1 is achievable through the computation outlined in Equation 2.2:

$$\mathbf{x} = \lfloor \frac{\mathbf{Sc}}{w} \rfloor \tag{2.2}$$

where $\lfloor \cdot \rfloor$ means rounding the number down to its nearest integer.

Considering the three basic operations on integer vectors - addition, multiplication, and weighted inner products that are readily executable within the encryption scheme detailed in [32], it becomes feasible to efficiently compute any polynomial on integers within the DNN.

Consider two integer vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$, for which three fundamental operations are defined:

I Addition, $\mathbf{x}_1 + \mathbf{x}_2$, necessitating equal lengths for $\mathbf{x}_1$.
II Linear transformation, $\mathbf{G}\mathbf{x}_1$ utilizing an arbitrary matrix $\mathbf{G}$.
III Weighted inner products, $\{\mathbf{x}_1^T \mathbf{H}_j \mathbf{x}_2\}$, employing a set of weight matrices.

It is presumed that all values resulting from these operations fall within zero and $\lfloor \frac{q}{w} \rfloor$, ensuring the absence of integer overflows. Additionally, let $\mathbf{c}_1$ and $\mathbf{c}_2$ represent the ciphertexts corresponding to $\mathbf{x}_1$ and $\mathbf{x}_2$ with secret keys $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, and they adhere to:

$$\mathbf{S}_i \mathbf{c}_i = qk_i + w\mathbf{x}_i + \mathbf{e}_i \tag{2.3}$$

with $|\mathbf{S}_i|$, $|\mathbf{k}_i|$ and $|\mathbf{e}_i|$ much smaller than q. Here, we show the three types of fundamental operations, referring from the paper [32] for the demonstration.

I. Addition Operation: if $\mathbf{c}_1$ and $\mathbf{c}_2$ have the same secret key, i.e. $\mathbf{S}_1 = \mathbf{S}_2 = \mathbf{S}$, then $\mathbf{c}' = \mathbf{c}_1 + \mathbf{c}_2 \mod q$ is an encryption of $\mathbf{x}_1 + \mathbf{x}_2$, since

$$\mathbf{Sc}' = q\mathbf{k}' + w(\mathbf{x}_1 + \mathbf{x}_2) + (\mathbf{e}_1 + \mathbf{e}_2) \tag{2.4}$$

Given that $\mathbf{k}'$ is an integer vector of small magnitude, and assuming $\mathbf{x}_1$ and $\mathbf{x}_2$ may utilize different secret keys, our initial step involves aligning one secret key with the other using key-switching techniques, for instance, changing $\mathbf{S}_1$ to $\mathbf{S} = \mathbf{S}_2$. Alternatively, if $\mathbf{S}_1$ and $\mathbf{S}_2$ are correlated, it becomes essential to switch both to a common key $\mathbf{S}$. Let us denote $\mathbf{c}'_1$ and $\mathbf{c}'_2$ as the ciphertexts resulting from key-switching. These ciphertexts fulfill the condition $\mathbf{Sc}'_i = q\mathbf{k}'_i + w\mathbf{x}_i + \mathbf{e}'$, where both $|\mathbf{k}'|$ and $|\mathbf{e}'|$ are small. If we set $\mathbf{c}' = \mathbf{c}'_1 + \mathbf{c}'_2 \mod q$, then

$$\mathbf{Sc}' = q\mathbf{k}' + w(\mathbf{x}_1 + \mathbf{x}_2) + \mathbf{e}' \tag{2.5}$$

It is noted that $|\mathbf{k}'| \leq |\mathbf{k}'_1| + |\mathbf{k}'_2|$ and $\mathbf{e}' = \mathbf{e}'_1 + \mathbf{e}'_2$.

II. Linear Transformation: The linear transformation $\mathbf{Gx}_1$ follows the observation that

$$\mathbf{GSc}_1 = q\mathbf{Gk}_1 + w\mathbf{Gx}_1 + \mathbf{Ge}_1. \tag{2.6}$$

Therefore, if the magnitude of $|\mathbf{G}|$ is significantly smaller than q, $\mathbf{c}' = \mathbf{c}_1$ can be considered as the ciphertext representing $\mathbf{Gx}_1$, under the secret key $\mathbf{S}' = \mathbf{GS}$.

III. Weighted Inner Products: A set of weighted inner products $\mathbf{x_1^T H_j x_2}$ can be calculated using the method for multiplication through tensor products. The corresponding formula is

$$\mathbf{s}'_j c' = q\mathbf{k}'_j + w(\mathbf{x}_1^T \mathbf{H}_j \mathbf{x}_2) + \mathbf{e}'_j \tag{2.7}$$

where $\mathbf{s}'_j = vec(\mathbf{S}_1^T \mathbf{H}_j \mathbf{S}_2)^T$ be the jth row of the new secret key $\mathbf{S}'$, let $\mathbf{c}' = \lceil \frac{vec(\mathbf{c}_1 \mathbf{c}_2^T)}{w} \rfloor_q$ be the new ciphertext, and $\mathbf{k}'_j$ is an integer, and $\mathbf{k}'_j, \mathbf{e}'_j$ are much smaller than w.

Key-switching Technique.

In their groundbreaking study, Brakerski and Vaikuntanathan [36] presented a crucial re-linearization method that can modify the secret key in Process-level Virtualization

Machines (PVM) schemes to switch between various vector forms. Subsequently, Brakerski, Gentry, and Halevi extended this innovation, developing a method for transitioning between two matrix-form secret keys [37]. Brakerski and Vaikuntanathan's re-linearization technique generally involves two stages. It is utilized to transition a secret key $\mathbf{S} \in \mathbb{Z}q^{m\times n}$ to a different secret key $\mathbf{S}^{'} \in \mathbb{Z}q^{m\times n'}$, using an intermediate key $\mathbf{S}^{*} \in \mathbb{Z}_q^{m\times nl}$. This process is executed as follows, and in doing so, we also obtain a new ciphertext $\mathbf{c}'$ which continues to encrypt the same integer vector $\mathbf{x}$.

- Step 1: $\mathbf{S} \rightarrow \mathbf{S}^{*}$, indicating that $\mathbf{S}$ is transformed to $\mathbf{S}^{*}$, such that its corresponding new ciphertext $\mathbf{c}^{*}$ has a smaller magnitude than $\mathbf{c}$. The goal is to represent each element $\mathbf{c}_i$ in $\mathbf{c}$ with a binary vector or binary representation. Hence it results in a new ciphertext $\mathbf{c}^{*}$ with $|\mathbf{c}^{*}| = 1$. Assuming that $\mathbf{c}_i = \mathbf{b}_{i0} + \mathbf{b}_{i1}2 + \cdots + \mathbf{b}_{i(l-1)}2^{l-1}$, then we obtain $\mathbf{c}^{*}$ by expressing each $\mathbf{c}_i$ as $[\mathbf{b}_{i0}, \mathbf{b}_{i1}, \cdots, \mathbf{b}_{i(l-1)}]$. Then, we construct a secret key $\mathbf{S}^{*} \in 2^{m\times nl}$ such that

$$\mathbf{S}^{*}\mathbf{c}^{*} = \mathbf{S}\mathbf{c} \tag{2.8}$$

  it can be replaced by each $\mathbf{S}_{ij}$ in $\mathbf{S}$ with a vector $[\mathbf{S}_{ij}, \mathbf{S}_{ij}2, \cdots, \mathbf{S}_{ij}2^{l-1}]$.
- Step 2: $\mathbf{S}^{*} \rightarrow \mathbf{S}'$, showing that $\mathbf{S}^{*}$ is transformed to $\mathbf{S}'$. We construct an integer matrix $\mathbf{M} \in \mathbb{Z}^{n'\times nl}$ and a noise matrix $\mathbf{E}$, such that

$$\mathbf{S}^{'}\mathbf{M} = \mathbf{S}^{*} + \mathbf{E} \tag{2.9}$$

  assume $\mathbf{S}^{'} = [\mathbf{I}, \mathbf{T}]$ with an identity matrix $\mathbf{I}$, $\mathbf{M}$ can be constructed by:

$$\mathbf{M} = \left(\frac{\mathbf{S}^{*} + \mathbf{E} - \mathbf{T}\mathbf{A}^{*}}{\mathbf{A}}\right) \tag{2.10}$$

  where $\mathbf{A} \in \mathbb{Z}^{(n'-m)\times nl}$ is a random matrix.
- Step 3: $\mathbf{S}', \mathbf{M} \rightarrow \mathbf{c}'$, indicating that we obtain a new ciphertext $\mathbf{c}'$ by $\mathbf{M}$:

$$\mathbf{c}' = \mathbf{M}\mathbf{c}^{*} \tag{2.11}$$

Algorithm 1. Efficient Integer Vector Homomorphic Encryption Scheme (EIVHE)

Input: $w, \mathbf{x}$

Output: $\mathbf{c}, \mathbf{S}$

Get the row of $\mathbf{x}$ m and the column of $\mathbf{x}$ n, respectively;
Generate $key() : \mathbf{S} = random.rand(m, n) \times w$;
Get $T() : \mathbf{T} = random.rand(n, 1)$;
Get l: encode length for a maximum integer in $|\mathbf{x}|$;
Get $\mathbf{c}^*$;
for $i = 1$ to m
    for $j = 1$ to n
        $\mathbf{b} = binary\_vector(\mathbf{c}[i][j])$;
        if $\mathbf{c}$[i][j] ≤ 0 then $\mathbf{b}^* = -1$;
        $\mathbf{c}^*[i][(j * 1) + l - len(\mathbf{b}) : (j + 1) * l] += \mathbf{b}$;
    end for
end for
Get $\mathbf{S}^*$;
for $i = 1$ to l;
    $\mathbf{S}^* = \mathbf{S} * 2^{l-i-1}$;
end for
$\mathbf{S}^{'} = [\mathbf{I}, \mathbf{T}]$ and $\mathbf{A} = random.rand(1, n * l) * 10$;
$\mathbf{E} = random.rand(\text{the row of } \mathbf{S}^*, \text{the column of } \mathbf{S}^*) * l$;
$\mathbf{M} = \begin{pmatrix} -\mathbf{TA} + \mathbf{S}^* + \mathbf{E} \\ \mathbf{A} \end{pmatrix}$;
$\mathbf{c}' = \mathbf{M} * \mathbf{T}$;
Return $\mathbf{c} = \mathbf{c}'$ and $\mathbf{S} = \mathbf{S}'$;

2.4.2 Constructed Neural Network Model

We construct six hidden layers to prevent gradient dispersion and use batch normalization. The activation function is Rectified Linear Unit (ReLU), and the output layer is the SoftMax function [45] to do classification, which is approximated and

transformed into the polynomials. Moreover, the cost function is cross-entropy, and the optimization function is Adaptive moment estimation (Adam) [46]. To train the DNN, assume **X** denotes the training dataset, **X_label** denotes the label of **X**, **Y** denotes the test dataset, and **Y_label** denotes the label of **Y**, then feed them into a DNN model, so we propose a DL algorithm in Algorithm 2. We explain the functions in the following:

Batch Normalization

According to the work [47], at each Stochastic Gradient Descent (SGD), the corresponding activation is normalized by mini-batch so that the mean value of the result, the output signal of each dimension, is zero and the variance is one.

ReLU

The ReLU is an activation function that is commonly used in the DNNs. In general, the linear rectification function refers to the slope function in mathematics [48]; the ReLU is approximated as the polynomial function, such as:

$$f(X) = max(x, 0) \rightarrow f(X) = x^2 \tag{2.12}$$

SoftMax

The SoftMax is a generalization of a logistic function that "squashes" or maps a K-dimensional vector $\mathbf{Z}$, which compresses a K-dimensional vector of an arbitrary natural number into another K-dimensional real vector, the values of each element of the vector fall in the range $(0, 1)$ and adds up to one.

$$\sigma(\mathbf{Z})_j = \frac{e^{\mathbf{Z}_j}}{\sum_{k=1}^{K} e^{\mathbf{Z}_k}} \tag{2.13}$$

The SoftMax function defined in Equation 2.13 inherently involves exponential operations incompatible with the HE due to their computational complexity and non-polynomial nature. Consequently, this function cannot be directly applied to the HE in its existing form. To solve this problem, it is necessary to approximate Equation 2.13 with polynomials.

In this study, a naive approximation is made by expanding both the numerator and denominator using a Taylor series, truncating after the quadratic term. Although the division process does not always result in a polynomial since the HE schemes often operate in a ring of integers modulo a large prime or a power of a prime, especially when the remainder is nonzero, we assume here that the remainder is sufficiently negligible, thereby treating the division result as an approximate polynomial representation of the softmax function. However, this assumption has not been theoretically validated, indicating a crucial area for further investigation. As the following sections will show, the limited accuracy achieved in this study is likely due to these naive approximations, highlighting the need for future theoretical exploration and verification in this area.

Cross-entropy

Cross-entropy is a concept of the information theory, the earliest by the information entropy change, which processes relative information entropy and compression ratio, and then be used in many places, including communications, error correction circuits, game theory, machine learning, etc. [49]. The introduction of a cross-entropy function for the DNN is to make up for the defect that the derivative form of the Sigmoid function is easy to saturate, as Equation 2.14:

$$L_H(x, z) = -\sum_{k=1}^{d} x_k \log zx + (1 - x_k) \log(1 - z_k) \tag{2.14}$$

Adam

The Adam [46] is a first-order gradient-based algorithm that aims at optimizing a random objective function. According to the first-order moment estimation and second-order moment estimation of the gradient of each parameter via cost function, Adam dynamically adjusts the learning rate of each parameter.

Algorithm 2. Deep Learning with the EIVHE

Input: training data X, labels of training data X_label, validate data Y, labels of validate data Y_label

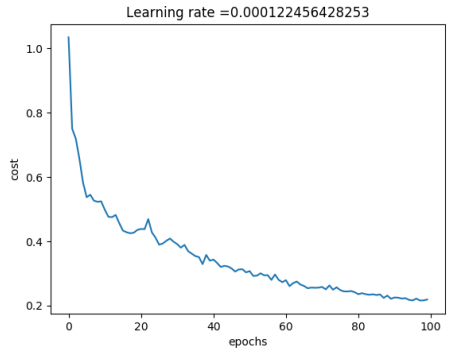
Output: prediction accuracy p

1: Initial weight w and parameters par of the DNN
2: Feed training data X into the DNN→ EIVHE(X)
3: Get mini-batch $batch_x$
4: Forward propagation: computing the cost
5: Backward propagation: computing gradients
6: Optimization: the Adam
7: Repeat steps 3 − 7 until the DNN becomes stable
8: Get the well-trained model
9: Input validate data Y into the well-trained DNN model→ EIVHE(Y)
10: Return p

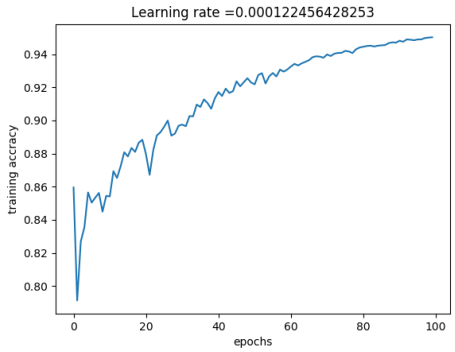
2.5 Experiments

2.5.1 Dataset

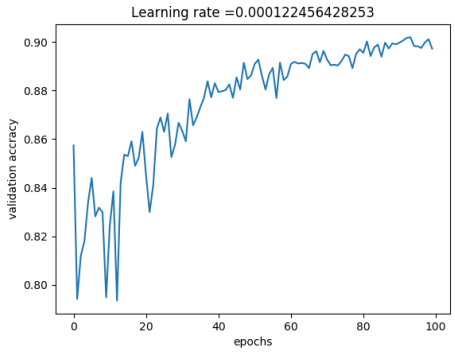
We conduct experiments based on the MNIST dataset for handwritten digit recognition consisting of 60, 000 training examples and 10, 000 test examples [43]. Each example is a 28×28 size gray-level image. One image has 784 feature points before the HE, and we get 785 feature points per image using this method. In other words, according to the key-switching technique, through matrix transformation, the feature point of one image has changed from 784 to 785. We use a simple forward-propagation DNN with the ReLU, SoftMax of 10 classes with cross-entropy loss, mini-batch normalization, and the optimization algorithm Adam. We use our scheme to apply to the MNIST and do three representative experiments on this dataset.



(a). Cost of model



(b). Training accuracy of model



(c). Validation accuracy of model

Figure 2.1. Cost and accuracy of the original MNIST dataset in 100 epochs

2.5.2 Baseline Model

Our baseline model uses the MNIST dataset without processing, where the input layer has 784 units, six hidden layers with 512, 256, 128, 64, 32, and 16 units, respectively, and the output layer has 10 units. As for the Adam, we set the parameters as default: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Using this method, we can reach an accuracy of 94.87% in about 100 epochs.

As shown in Figure 2.1a, the cost of the original dataset maintains a downward trend until convergence. Figure 2.1b shows the trend of training accuracy until convergence, and it arrives at the training accuracy of 94.87%. Figure 2.1c shows the validation accuracy of the original test dataset, and the convergence value is 90.19%. Details are shown in Table 2.1.

Table 2.1. Details of original dataset

Type	Training Accuracy (%)	Validation Accuracy (%)	Training Time (minute)	Epoch
Original Dataset	94.87	90.19	26.28	100

2.5.3 Proposed Model

We test the proposed DL with the HE model with the same architecture as the baseline model. It is different because we use the encrypted dataset to feed the DNN. The original size of one image of the MNIST is 28×28, which is reshaped as 1×784. Thus, training images become a $60,000 \times 784$ matrix, and test images become a $10,000 \times 784$ matrix. More than that, We use one-hot encoding to differentiate the training and test labels. The encryption time of the two datasets is shown in Table 2.2.

Table 2.2. Homomorphic Encryption Time

Dataset	Encryption Time (second)	Cost per Pixel (second)
Training images	662.77	1.41×10^{-5}
Test images	119.98	1.53×10^{-5}

2.6 Performance Analysis

2.6.1 Accuracy on the Separate Datasets

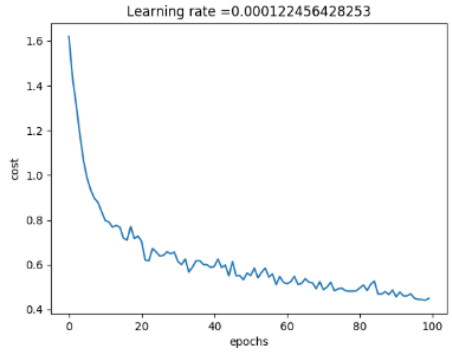
At first, we encrypt training image datasets and test image datasets, respectively, as shown in Algorithm 1. Then, we feed the encrypted training image datasets into the DNN model, where the epoch is 100. Under encryption without using the same key, the matrix we get is a non-homomorphic matrix that somewhat undermines the internal relationship of the datasets. Then, to solve this issue, we let two datasets encrypt together with the same key as shown in Section 2.4.2.

As shown in Figure 2.2a, the cost is much higher than the original dataset. Furthermore, Figure 2.2b shows the training accuracy of the encrypted dataset is 85.97%. We think the reason for the decrease in accuracy is that after encryption, our input dimension changed from 784 to 785. It seems like we artificially added one fake feature point in every input picture, which can be regarded as a type of noise. However, as shown in Figure 2.2c, we check out that our validation accuracy is greatly low. By checking the dataset, we realize that the fatal point is that we encrypt training and test datasets, respectively, but each encryption secret key is randomly generated. Details are shown in Table 2.3.

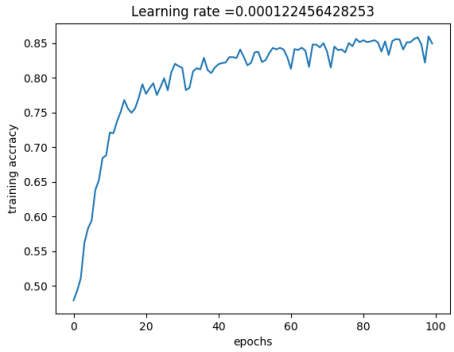
2.6.2 Accuracy on the Improved Algorithm

To show a better performance, we improve Algorithm 1 as shown in the new Algorithm 3-Improved EIVHE (IEIVHE).

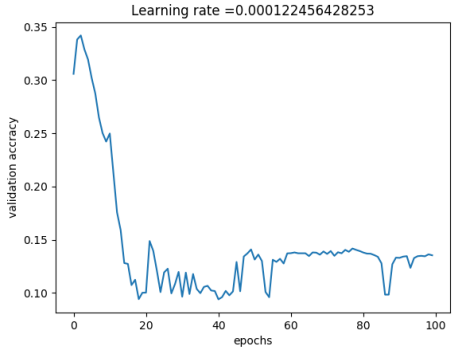
Thus, we use the IEIVHE to encrypt our training and test datasets together, and



(a). Cost of model

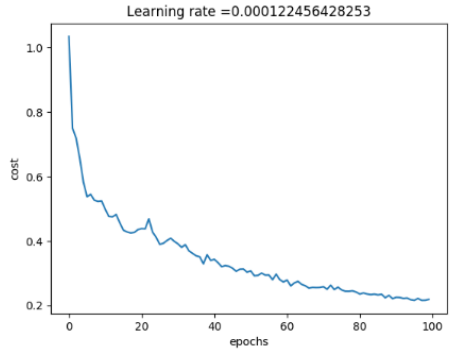


(b). Training accuracy of model

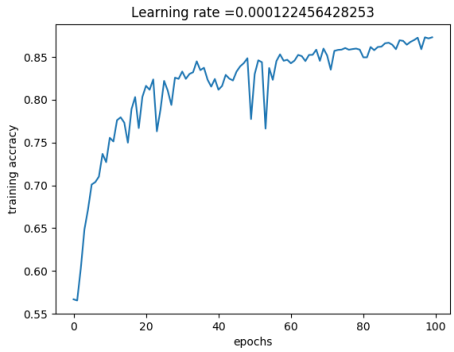


(c). Validation accuracy of model

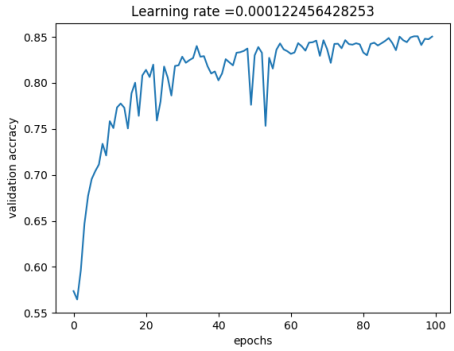
Figure 2.2. Cost and accuracy of the separate encrypted MNIST dataset in 100 epochs.



(a). Cost of 100 epochs



(b). Training accuracy of 100 epochs



(c). Validation accuracy of 100 epochs

Figure 2.3. Cost and accuracy of the encrypted MNIST dataset with the improved algorithm in 100 epochs.

Table 2.3. Accuracy of the separate encrypted dataset

Type	Training Accuracy (%)	Validation Accuracy (%)	Training Time (minute)	Epoch
Encrypted Dataset (respectively)	85.97	13.54	28.38	100

then we can get better results. As shown in Figure 2.3a, we find out that our cost has declined, lower than the original dataset. In Figure 2.3b, we learn that our training accuracy has improved to 87.29%, 1.32% higher than the original datasets. Figure 2.3c shows that the validation accuracy has improved dramatically as 85.04%. Details are shown in Table 2.4.

Algorithm 3. Improved efficient integer vector homomorphic encryption (IEIVHE)

Require: $w, \mathbf{X}, \mathbf{Y}$;

Ensure: $\mathbf{c_1}, \mathbf{c_2}, \mathbf{S_1}, \mathbf{S_2}$;

1: $m = \mathbf{X}.shape[0], n = \mathbf{X}.shape[1] = \mathbf{Y}.shape[1], h = \mathbf{Y}.shape[0]$;
2: **S**=generate_key(w, n);
3: **T**=get_T(n);
4: $\mathbf{c_1}, \mathbf{S_1}$=encrypt_via_switch(**X**, w, m, n, **T**, **S**);
5: $\mathbf{c_2}, \mathbf{S_2}$=encrypt_via_switch(**Y**, w, h, n, **T**, **S**);

Table 2.4. Accuracy of the improved algorithm on encrypted dataset

Type	Training Accuracy (%)	Validation Accuracy (%)	Training Time (minute)	Epoch
Encrypted Dataset	87.29	85.04	28.08	100

As for the accuracy improvement, we have made many attempts at hyper-parameter tuning. Unfortunately, so many hyper-parameters are fixed, which are widely circulated via the industry, so we set the recommended parameters as the default. On the other hand, there is one parameter that still has a chance to impact the accuracy

of the model: epochs. Following this clue, we change the epochs for accuracy improvement. Details are shown in Table 2.5.

Table 2.5. Accuracy of the impact of epochs

Type	Training Accuracy (%)	Validation Accuracy (%)	Training Time (minute)	Epoch
Encrypted Dataset	87.29	83.38	55.59	200
	87.28	83.39	85.97	300

According to Table 2.5, as the number of epochs increases, the accuracy almost arrives at a convergence level at 200 epochs, around 87.28%.

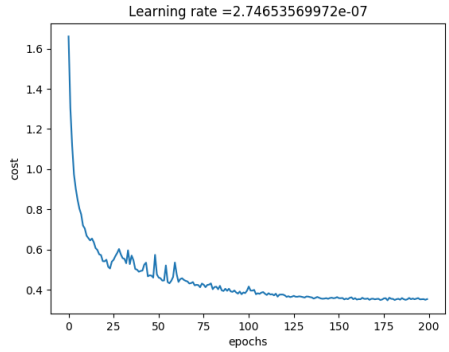
2.6.3 Accuracy on the Absolute Datasets

Besides accuracy improvement from the level of hyper-parameters, we try to discern a new way to improve it. Through internal data analysis, we discover that the encrypted data matrix alters from a non-negative matrix to a non-positive matrix. In our DNN model, the active function is the ReLU. This causes most neurons to be inactive. Hence, a delicate and easy method is devised to solve the issue that takes the absolute value of the encrypted matrix.

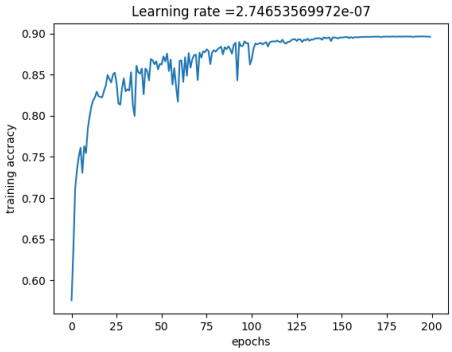
As shown in Figure 2.4a, the cost is roughly the same as above. Figure 2.4b shows improved accuracy, which means our method has an effect and improved slightly. Then, as for Figure 2.4c, we can see that the validation accuracy has also improved. Details are shown in Table 2.6.

From the above experiments, we observe the following:

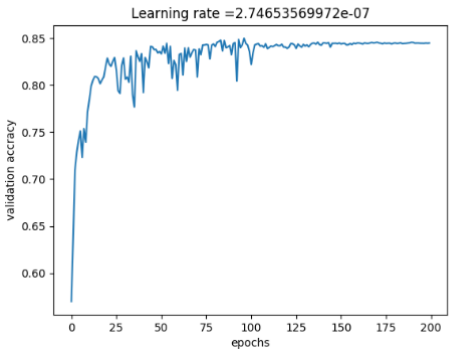
- The HE protects the privacy of data. We can encrypt the dataset before using it via neural networks. However, we need a lot of time to encrypt the dataset. It just takes some time to encrypt the training dataset and to train our model, and then every picture that needs to be encrypted takes a very short time to encrypt.



(a). Cost of 200 epochs



(b). Training accuracy of 200 epochs



(c). Validation accuracy of 200 epochs

Figure 2.4. Cost and accuracy of the absolute encrypted MNIST dataset with the improved algorithm in 200 epochs.

- When we take the absolute value of the encrypted dataset, accuracy has improved 3.08% via the level of data.
- The training super-parameters have little impact on the model accuracy. As for epochs, we get a stable and efficient value of 200.

Our algorithm allows for the PPDL. The first initial experiments did not show a significant improvement, but it is interesting to consider more sophisticated schemes for accuracy improvement.

Table 2.6. Accuracy of the absolute datasets

Type	Training Accuracy (%)	Validation Accuracy (%)	Training Time (minute)	Epoch
Encrypted Dataset (absolute)	89.05	84.98	256.73	200

2.7 Conclusion

We demonstrate the training of the DNN model with encrypted datasets of the HE, preserving the user's privacy and avoiding privacy loss when using a neural network model computed over an entire model with many parameters. In our experiments for the MNIST, we achieved 89.05% training accuracy with the encrypted dataset with absolute value in the IEIVHE. Our scheme is based on a dataset of the HE; after encryption, we use the encrypted dataset to train our DNN model. Since our approach applies cryptography to the DL, it can be adapted to many other datasets before using them to feed. Others cannot understand data without a secret key when conducting data encryption. Thus, it can protect privacy and ensure users use the neural network safely. However, it is recommended that accuracy will decline when data is encrypted. The reason significantly is the noise. After encryption, the dataset's features will increase; we can call these features noise, which are not fundamental features in datasets. This chapter demonstrates a specific procedure for applying EIVHE to neural networks by combining techniques such as key-switching,

batch normalization, and matrix transformations. Then, it evaluates its performance in the EIVHE and proposed IEIVHE.

Chapter 3

Contribution 2: Channel-wise Homomorphic Encryption for Ciphertext Inference in Convolutional Neural Network [*1]

3.1 Introduction

Users upload data to cloud servers to make use of a service such as a training Machine Learning (ML) model. These are known as Machine Learning as a Service (MLaaS) [50], especially implementing Deep Learning (DL) for the platform set as DL as a Service (DLaaS). However, the users can be reluctant to upload their personal and sensitive data as the server might not be secure. Presently, servers leverage encryption to protect the privacy of user data. Homomorphic Encryption (HE) [5], which can perform arithmetic computations on ciphertext, is one of the promising Privacy-preserving DL (PPDL) [3] methods.

[*1] This chapter is based on the paper: [CHE: Channel-Wise Homomorphic Encryption for Ciphertext Inference in Convolutional Neural Network, IEEE Access, 2022].

Although there have been several studies [6] [7] [8] [9] [10] [11] [12] on HE-based PPDL, the achieved latency was not short enough, and the accuracy was not high enough. Gilad et al. [6] developed ciphertext inference of Modified National Institute of Standards and Technology (MNIST) [51] dataset, but the accuracy was 96%, and the latency was 570 seconds. Furthermore, Ishiyama et al. [9] achieved an accuracy of 99.18% and a latency of 21.15 seconds on the MNIST. We aim to improve the performance by targeting image classification of the HE-based PPDL by combining two approaches: Channel-wise Homomorphic Encryption (CHE) [52], and Batch Normalization (BN) [47] with Coefficient Merging (CM) [53].

Most previous studies [6] [9] [11] have adopted Pixel-wise Homomorphic Encryption (PiHE) [54] for image classification, where multiple images are packed together and processed at once. However, we argue that this approach may not be suitable for all practical applications as the user may want the DLaaS to process only one or a few images, not a batch of images. To implement empirical application and improve processing efficiency, we aim to leverage the CHE, which provides shorter inference latency because it processes the data in a channel instead of in a pixel. It packs elements of one channel into a single ciphertext, forming a vector, which can perform Single Instruction Multiple Data (SIMD) [55] processing.

Aharoni et al. [56] introduced a new packing-oblivious programming framework, which somewhat generalized the CHE and PiHE. Although Dathathri et al. [57] and Lou et al. [58] have previously proposed a conceptually similar CHE judged from the provided figures, the algorithm was not described in detail. The BN layer is widely used in the neural network (NN) [59] model to achieve high accuracy [47], whitening the input images via the shift and the bias. Chabanne et al. [7] placed the BN layer before the activation layer to attain a restricted stable distribution at the entry of Rectified Linear Unit (ReLU) [60]. HEMET [58] set the BN layer behind the activation layer and obtained excellent performance. Moreover, Ishiyama et al. [9] placed the BN layer before the activation layer. However, there is still insufficient evidence to decide if placing the activation layer before the BN layer is better or vice versa. Given this background, we formulate several algorithms and one computation scheme, "Onion," to achieve ciphertext inference in Convolutional Neural Network

(CNN) [61] through the CHE approach.

To address previous problems, we point out the two schemes with different orders of the convolution layer, activation layer, and BN layer, namely, Convolution-Activation-Batch (CAB) and Convolution-Batch-Activation (CBA) for utilizing the MNIST and Canadian Institute for Advanced Research-10 classes (CIFAR-10) [62] datasets. Each scheme is a method of the CM, and mathematical formulae are given to demonstrate the scheme's stability later. The latency is related to the number of predefined levels, so naively deploying the BN layer increases multiplicative depth (MD) [63]; since the number of the HE multiplications and rotations are highly related to the execution time, which exploits a higher level, it leads to longer latency. We fuse the BN and activation layer with the CM as a mapping layer, which increases accuracy, decreases the latency of ciphertext inference, and describes the functions of the two schemes. In this chapter, we leverage the symbol "[.]" for ciphertext to distinguish between plaintext and ciphertext more clearly.

Our contributions are summarized as follows:

- We propose the explicit algorithms of the CHE to address the HE-based ciphertext inference problem in the CNN for higher accuracy and shorter latency.
- We formulate the BN layer with the CM in the CAB and CBA schemes and show the derivation of mathematical formulae. The two schemes can reduce ciphertext inference latency without increasing the MD.
- We evaluate and compare the proposed CHE against five works [6] [7] [8] [9] [11], and achieve 99.30% of accuracy and 7.76 seconds of latency on the MNIST, 76.40% of accuracy and 111.91 seconds of latency on the CIFAR-10, the result verifies that the CHE is practical to the end-user during the DLaaS.

3.2 Background

3.2.1 Homomorphic Encryption

The HE is an asymmetric cryptographic method that facilitates arithmetic computations over encrypted data or ciphertext. Since the HE operations introduce a certain amount of noise into the ciphertext, when the accumulated noise grows beyond a noise budget, decryption of the HE does not resolve the correct result [10]. Although a bootstrapping operation proposed by Gentry [38] could reduce the accumulated noise in a ciphertext, it is too time-consuming for practical applications. To address this limitation, Cheon et al. [64] proposed the approximate HE with Residue Number System (RNS) [65] and named the RNS-variant as Cheon-Kim-Kim-Song (RNS-CKKS) HE scheme. It is a Leveled HE (LHE) scheme, setting a threshold for noise budget to compute finite HE operations without bootstrapping. The RNS-CKKS HE scheme denotes the degree of polynomial modulus by N, which is a power of two, encodes data to $N/2$ fixed-point numbers, if not enough padding with zero, and then encrypts $N/2$ numbers into $N/2$ slots of one ciphertext.

The following are the core algorithms of the RNS-CKKS HE scheme:

I Encode operation encodes the input vector into the polynomials for encryption.

II The HE operations include HE addition, HE multiplication, and HE rotation.
   i The HE addition makes element-wise addition for slots in the corresponding position.
   ii The HE multiplication performs element-wise multiplication.
   iii The HE rotation rotates the slots of ciphertext, similar to the left and right panning.

A HE operation is performed simultaneously on all slots of the ciphertext. A ciphertext is a polynomial of degree N with each coefficient representing modulo Q, where Q is a product of n primes, describing the level of the HE. To reduce the

noise, a rescaling operation is needed to convert n-level ciphertext to $(n-1)$-level ciphertext. The representative encode function and rotation function are shown as follows:

- $Encode(z) \longrightarrow p$: the CKKS exploits the rich structure of integer polynomial rings for its plaintext and ciphertext spaces. Nonetheless, data comes more often in the form of vectors than in polynomials. Encoding input vector z into a polynomial p is necessary.
- $Rot([c], n) \longrightarrow [c']$: Deploy linear rotation over ciphertext $[c]$, n is the step size, which cycles ciphertext horizontally to the left. n starts from one rather than zero, and this function needs one key, the Galois key, set as default.

3.2.2 Privacy-preserving Deep Learning

Security and privacy preservation of the data must be considered to convince the user to upload their data to a server. The HE allows data to be processed in ciphertext, maintaining a trade-off between privacy and performance. To protect the user's privacy, the PPDL is built to perform inference directly on encrypted data; the ciphertext inference can be made over two paradigms: interactive paradigm or online mode, such as GELU-NET [66] whose model was partitioned the NN into two non-colluding parties, GAZELLE [26] which has high accuracy but huge memory cost, and BAYHENN [67] whose inference was partitioned into linear and non-linear computations; and non-interactive paradigm or offline mode, such as CryptoNets [6] which is the first work developed in the PPDL, Chabanne et al. [7] which is the first proposed work that combines activation function with the BN, Hesamiford et al. [8] which suggested using derivative of polynomials to replace the activation function, Ishiyama et al. [9] which proposed using the CM for decreasing the MD, Xie et al. [68] which exploited the Efficient Integer Vector HE in the CNN and Dathathri et al. [57] which initiated an optimizing compiler for the NN inference of the HE.

In this work, the focus is on ciphertext inference in the non-interactive paradigm. Table 3.1 shows the differentiation and comparison of the PiHE and CHE. Although the CHE and PiHE obtain similar accuracy, the CHE can shorten the latency and

reduce the consumed memory, but it will smaller the throughput.

3.2.3 Threat Model

The threat model is referenced from the prior PPDL works [57] [58]. We assume that the machine learning server is a potential attacker, which is honest-but-curious, implying that the server does not deviate from the defined protocol but attempts to learn all the possible information from legitimately received messages [72]. The mission of privacy-preserving is to protect users' data while uploading to the cloud and making inferences using the well-trained model, which assumes there is a man-in-the-middle attack between the user and server and an adversary who tries to analyze data in the server.

To be more realistic and safer, instead of generating the public and private keys by the user, we assume that keys are distributed by a trusted third-party organization, which differs from previous threat models. The third-party certificate authority generates and distributes the user's public and private keys. A public key of the HE encrypts the data sent to the server and privacy-preserved by the encryption method. The model is trained using plaintext data in a safe, and parameters are stored in the server without revealing them to the user. The server accesses the plaintext parameters, which causes data leakage. It performs private and secure inference over encrypted data without decryption or accessing the secret key. Only the client decrypts the ciphertext result by the local stored secret key, as shown in Figure 3.1.

3.3 Related Work

The research in the PPDL area can be divided into five camps that implement different X-based methods: the HE-based, Secure Multi-Party Computation (SMPC)-based [73], Differential Privacy (DP)-based [74], Secure Enclaves (SE)-based [23], and Hybrid-based [75].

CryptoNets [6] is the first work to use the HE-based method to make ciphertext inference in the NN model. As CryptoNets became ineffective for the deeper NN, Chabanne et al. [7] designed and evaluated the first privacy-preserving classification

Table 3.1. Differentiation and comparison of PiHE and CHE. Note that assuming pixels and channels are less than or equal to the slot size.

Encryption	Related	Latency	Throghput	Memory	Number of Ciphertext
Pixel-wise	[6] [7] [8] [9] [11] [16] [66] [67] [68] [69] [70] [71]	Long	High	Larger	Number of pixels
Channel-wise	[10] [12] [26] [57] [58] CHE	Short	Low	Smaller	Number of channels

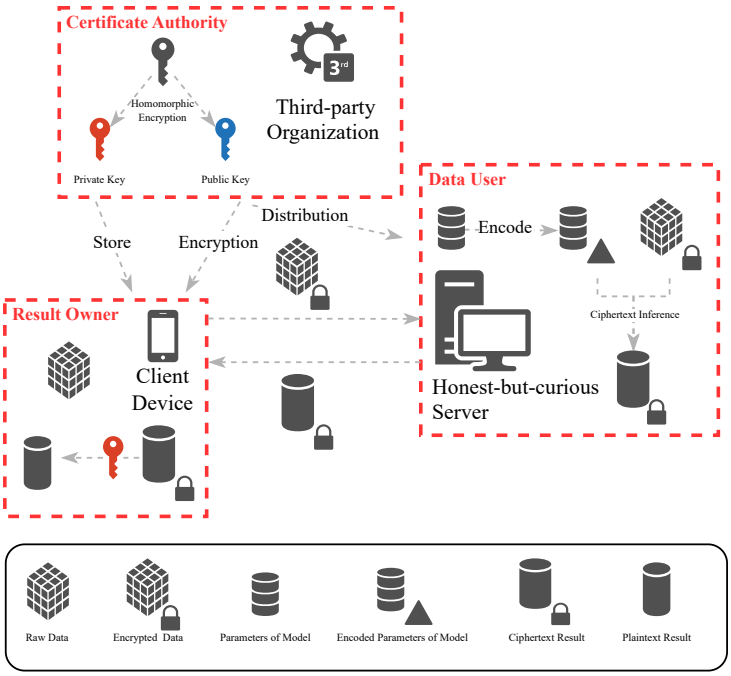


Figure 3.1. Threat model. There is no collusion. The server is honest-but-curious; it can access the plaintext machine learning model and perform ciphertext inference.

in the NN model. To protect privacy without accuracy loss, Juvekar et al. [26] and Zhang et al. [66] have proposed using the interactive paradigm to split the NN model into two parts. Furthermore, Hesamiford et al. [8] deployed the NN model on the server in the non-interactive paradigm, placing the activation function with the polynomials. Due to the time-consuming PPDL inference, Lou et al. [58] proposed implementing the mobile NN models to achieve shorter inference latency and higher inference accuracy. In TenSEAL [10], the authors implement the image-to-column (im2col) method to process the convolution layer. However, there is no way to find a simple way to build a network with more than one convolution layer, which is the non-interactive scheme.

The SMPC-based method exploited Yao's garbled circuits for secure two-party computation. Mohassel et al. [76] developed a combination of the garbled circuit with oblivious transfer and secret sharing. Rizai et al. [70] suggested a distributed training computation with a novel approach to secret sharing. Furthermore, Liu et

al. [71] devised a distributed SMPC framework for privacy-preserving data mining with one-hot encoding and a lower-upper decomposition algorithm.

The DP-based method adds random noise to a dataset and generates fake training data. PATE [16] originated a DP learning process utilizing teacher and student models. Fan et al. [77] initiated a local DP framework for data centers using the Laplacian mechanism to measure privacy-preserving quality.

The SE-based PPDL is the method by which a client sends data to the secure enclave environment where all the data and models are hidden from the outside world, such as [78]. The Hybrid-based PPDL is the method that leverages several techniques to build a secure and privacy-preserving model, such as GAZELLE [26].

Kim et al. [4] reported a comprehensive survey about the PPDL, showing a high-level view of the PPDL research.

3.4 Methodology

We propose a detailed and explicit CHE to run inference over encrypted data in the CNN and its workflow in Section 3.4.1. Moreover, we provide a high-level description of the BN layer with the CM in Section 3.4.2.

3.4.1 Algorithms of the CHE

In the proposed method, the image data is pre-processed channel-wise instead of pixel-wise. In practice, the pre-processing depends on the inference. Technically, the PiHE processes data in a matrix or tensor, while the CHE processes a vector; the direct difference is described in Figure 3.3. We introduce the data pre-processing scheme and functions of layers in the CNN by deploying the CHE. The details of different layers of methods over vector and tensor are shown in Table 3.2. The entire data processing schemes of the CNN model are demonstrated in Figure 3.2. It will be more comprehensible and readable to understand the following detailed designs and processes of the model, which uses the proposed CHE.

Table 3.2. Different functions of vector and tensor. Tensors are employed for the pixel-wise approach, and vectors are employed for the channel-wise approach.

Function	Data	
	Vector	Tensor
Convolution	Weighted sum between input vector(s) and filter(s) with the HE rotation	Calculate a cross-correlation between an input and a filter tensor
Element-wise operation	The same as "tensor" also includes activation with approximated polynomials	Such as the ReLU, BN, etc.
Pooling	Data is serial data, which has to simulate combining adjacent elements of tensor; also is the weighted sum between input tensor(s) and filter(s) without various weights, instead, fixed weights whose values equal to $\frac{1}{f*f}$; almost the same as "convolution."	Combine adjacent elements using a reduction operation such as maximum or average; also called down-sampling
Reshaping	Pack all outputs or ciphertexts of the last layer into one ciphertext preceding fully-connected layer	Reinterpret the shape of a tensor, for example, to flatten the preceding a matrix multiplication
Matrix multiplication	Since ciphertext is a vector, there is no way for matrix multiplication. Instead, it's vector-matrix multiplication which calculates vector or ciphertext and filter or plaintext matrix	Represent neurons that are connected to all inputs, which are found in dense layer typically at the end of a CNN

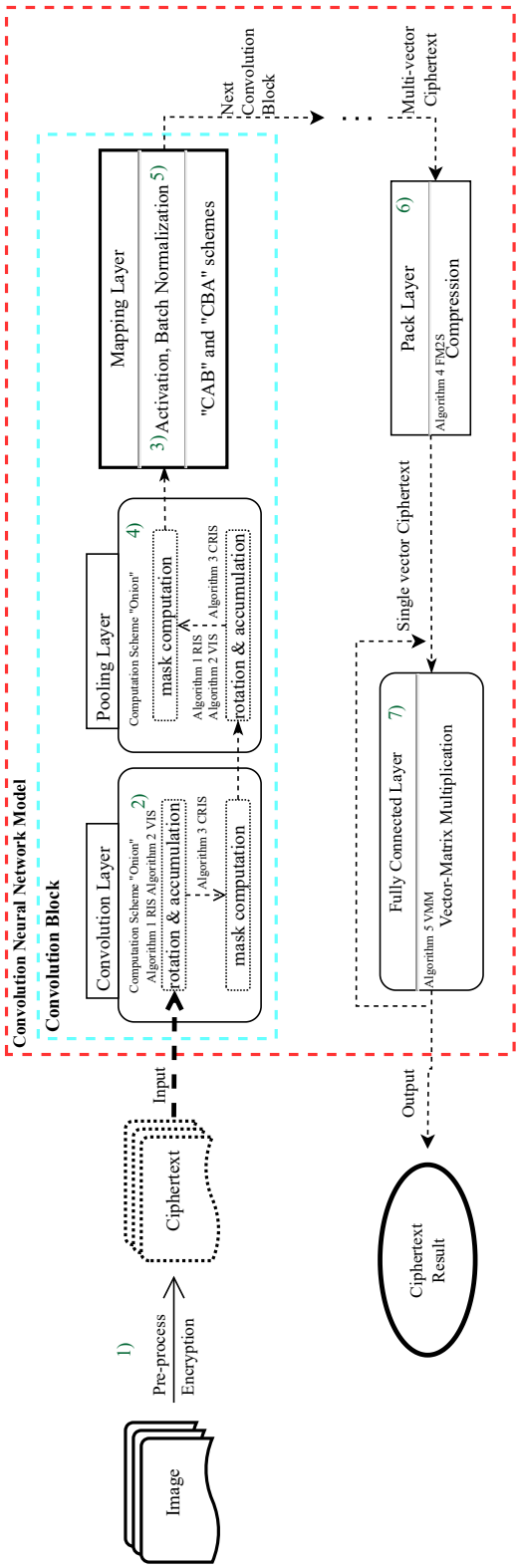


Figure 3.2. The flow chart showing the entire data processing schemes of the CNN model.

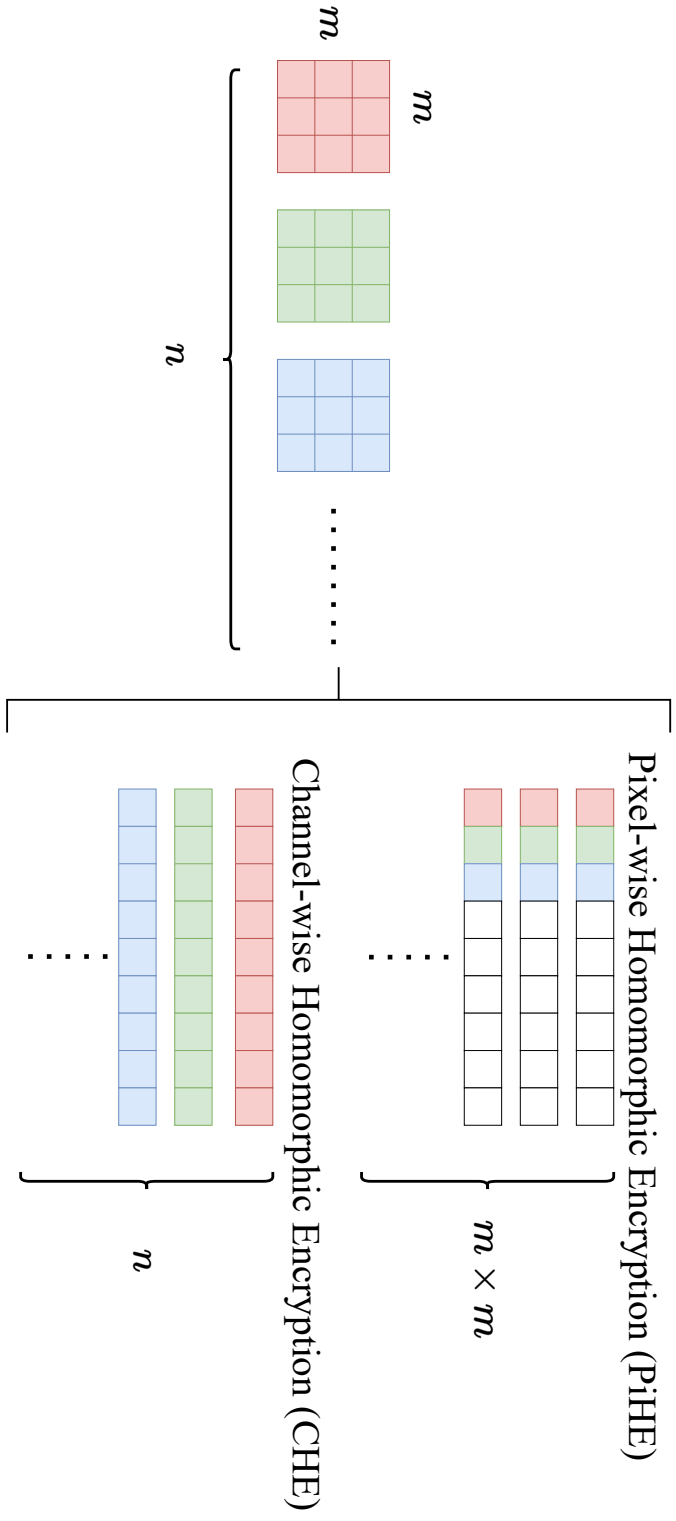


Figure 3.3. From the intuitive representation of the CHE and PiHE, it can be seen that using the PiHE yields $m \times n$ ciphertexts, whereas using the CHE only results in n ciphertexts.

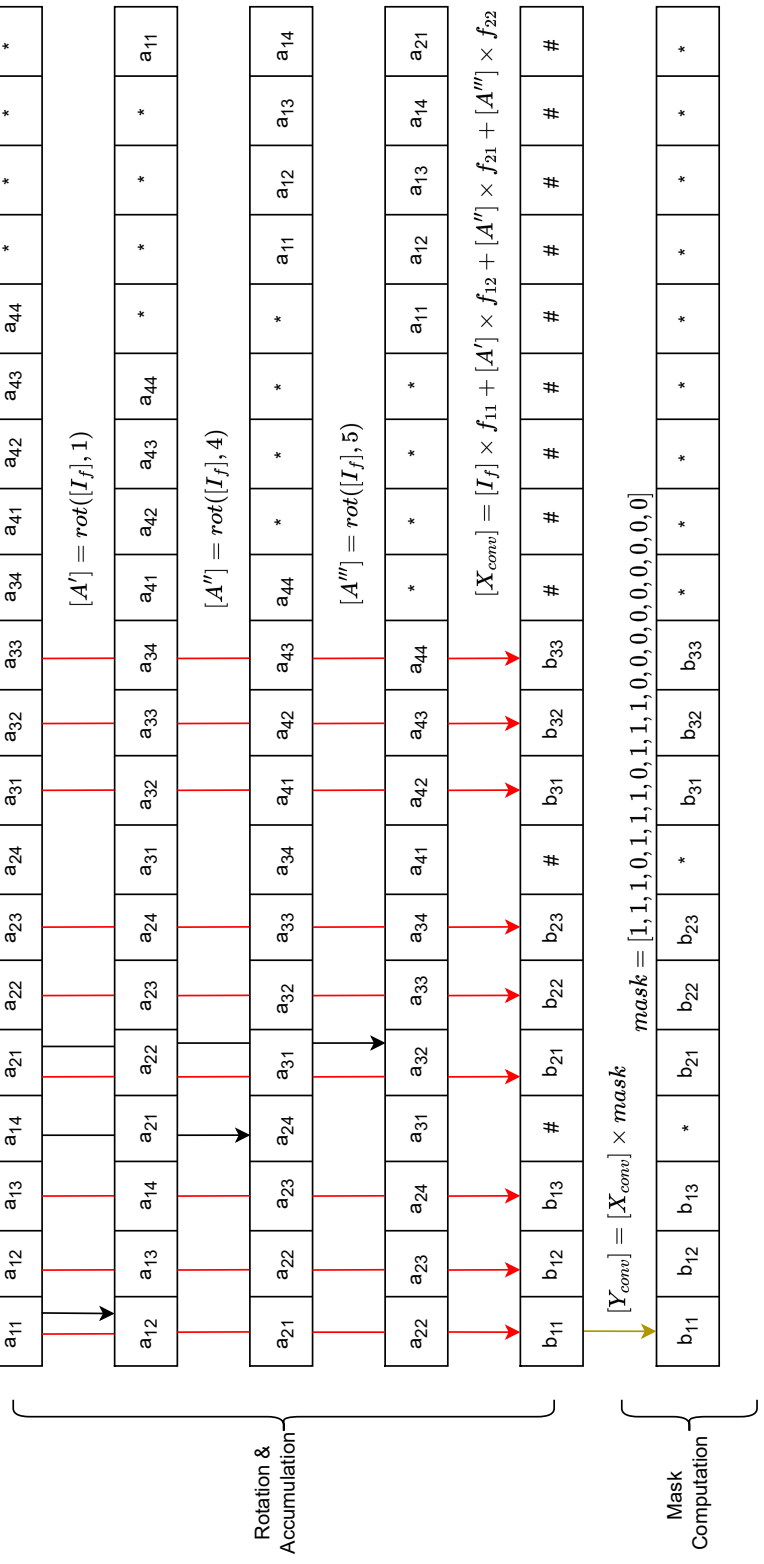


Figure 3.4. Convolution layer over ciphertext. Assume total slots are 20, the input size is 4×4, and the filter size is 2×2 with stride one. "#" means the redundant slot, and "*" means the slot holding zero or relatively small noise.

Figure 3.5. "Onion" computation scheme of the CHE shows how to calculate the ciphertext in a model without revealing a well-structured vector.

Pre-process and Encrypt Data

The image data consists of tensors from three channels. In the PiHE, the same pixels of multiple images are packed in a single ciphertext, and the number of ciphertexts equals the number of images. On the contrary, in the CHE, the first step is to flatten each channel of an image into a vector, then pack and encrypt each vector into one ciphertext, which is fed as the input to the NN model. In this case, the number of ciphertexts equals the number of channels of the image. For a single image, the CHE takes less time to transform the image into a ciphertext.

Reshape Convolution Layer

To deploy a deeper NN model with multi-convolution layers in a non-interactive scheme, we split the convolution layer into two parts: *rotation & accumulation*, and *mask computation*. Suppose the input has C channels, a height of H, and a width of W. It can then be encrypted into C ciphertexts, assuming that the product of H and W is less than or equal to the slot size $N/2$, which packs $H \times W$ input elements or pixels.

In *rotation & accumulation*, the HE rotation operation $Rot([c], n)$ is required to compute the element-wise HE multiplication between the input and weight filter, where $[c]$ is ciphertext and n represents the stride for rotation. Obtaining corresponding n values is an essential step of the convolution layer, which has not been described in previous works [57] [58]. We propose the algorithm Rotation Index Searching (RIS) to find n values, as depicted in Algorithm 4. Using a list of n values by the HE rotation operation $Rot([c], n)$, the model can calculate the element-wise HE multiplication between the input and each weight filter and accumulate the results into a single ciphertext. The result of a single ciphertext is a semi-manufactured output with two weaknesses: irrelevant slots in ciphertext, called noise, and chaotic structure of ciphertext compared with the input. In *mask computation*, a plaintext mask replaces the values of valid and irrelevant slots with one and zero, respectively, to remove noise. Note that the mask variable is a temporary vector with zeros and ones, where zeros represent redundant slots, and ones represent valid slots.

Next, the indexes of valid slots must be calculated to rearrange the chaotic struc-

ture. For this, we propose the algorithm Valid Index Searching (VIS) that is used to prepare the mask, recorded in Algorithm 5, and the return of the algorithm represents the indexes of ones.

After implementing *mask computation*, a well-structured vector, within valid values listed one after another, is necessary to maintain the exact structure of the input and output. We propose the algorithm Chaotic Rotation Index (CRIS) to look for valid values, as shown in Algorithm 6. However, *mask computation* is time-consuming and unsuitable for speedup inference. The method mentioned above of processing a chaotic vector for the next layer may not be necessary for the convolution layer, which we argue the model does not need to get the well-structured output. Instead, we propose a computation scheme, "Onion," shown in Figure 3.4, which relates to Algorithm 4 and Algorithm 5. The computation scheme Onion produces a chaotic vector as the output, which is a not well-structured vector of the convolution layer. The *rotation & accumulation*, which calculates with filter, and *mask computation*, which removes noise, cost the same as the one MD in ciphertext; thus, the MD of the convolution layer is two.

The description follows from Figure 3.4. Consider a 4×4 orange-colored matrix A filled with numbers from one to 16 left to right and top to bottom. In the first 2D convolution layer, a single 2×2 filter F with the stride $s(1, 1)$ samples the upper left corner element. After convolution, the ideal output is presented as numbers in a blue-colored solid matrix B, and the practical output is in a dotted matrix. Using the algorithm VIS, the model can get a list l_1 of indexes of valid values.

Similarly, the second layer is sampled by filter F. The output of this layer is presented in a purple-colored matrix C, and the index list l_2 is calculated by Algorithm 5. The objective of the model is to obtain the indexes of all the valid values. However, elements in l_2 cannot be used as subscripts of the output vector to find valid values. Instead, the elements in l_2 are regarded as subscripts in l_1, and the elements corresponding to these subscripts in l_1 then form the index set. The model leverages elements in the index set as the subscripts of the output vector. Hence, the model can find the required indexes, layer by layer, like an onion. Notably, the model must store an index list, passing and updating layer by layer, during the inference.

Algorithm 4. Rotation Index Searching (RIS)

Input: input feature map size m, convolution filter size f

Output: indexing list ris_idx

for $i = 0, ..., f - 1$ **do**
  Set line anchor $start \longleftarrow i \times m$
  for $j = 0, ..., f - 1$ **do**
    If $i = f - 1 \& j = f - 1$: continue
    Compute first point of line fp: $fp \leftarrow start + j$
    If $j = f - 1$:
      $ris\_idx_i \leftarrow (i + 1) \times m$
    Else: $ris\_idx_i \leftarrow fp + 1$
  end for
end for
return ris_idx

Algorithm 5. Valid Index Searching (VIS)

Input: input feature map size m, output feature map size n, layer's stride s

Output: indexing list vis_idx

for $i = 0, ..., n - 1$ **do**
  for $j = 0, ..., n - 1$ **do**
    $vis\_idx_i \leftarrow i \times s \times m + j \times s$
  end for
end for
return vis_idx

Activation Layer

The HE data only supports the polynomial function. Thus, the activation function should be approximated by a polynomial function to obtain high accuracy. The approximation method depends on the desired performance of the model, and there have been several studies [9] [69] [79]. As the method of approximating the polyno-

Algorithm 6. Chaotic Rotation Index Searching (CRIS)

Input: input feature map size m, output feature map size n, layer's stride s

Output: indexing list rot_idx

1: Indexing list $idx = VIS(m, n, s)$
2: **for** $i = 1, ..., n^2 - 1$ **do**
3: $\quad rot\_idx_i \leftarrow idx_i - i$
4: **end for**
5: **return** rot_idx

mial is not the objective of the current study, the square function is used to replace the ReLU. This square function is a second-order polynomial of the form ax^2+bx+c, where $a = 1, b = 0, c = 0$. The MD of the activation layer is one.

Pooling Layer

As max-pooling is a non-polynomial function, a polynomial function is needed to replace the max-pooling function. The HE scheme only supports multiplication and not division. Note that f is the size of the filter in the pooling layer. However, the division by $f \times f$ can be considered as multiplication by $1/f \times f$, as in Equation 3.1. Thus, to modify the max-pooling layer into the average pooling layer, we adopt the approach of multiplying the $1/f \times f$ the inverse of the number of the elements of the filter directly over all the elements of the filter. Note that the multiplication is performed over the ciphertext without decryption.

$$avg(X) = \frac{1}{f \times f} \sum_{i=1}^{f \times f} x_i \longrightarrow \sum_{i=1}^{f \times f} \frac{1}{f \times f} x_i \tag{3.1}$$

where the mask vector consists of zero and $\frac{1}{f \times f}$ and the MD of the average-pooling layer is one.

Batch Normalization

To ensure the stability of the NN training, it is necessary to carefully select the initialization method and choose a small value of learning rate, despite its increasing

complexity. To address the limitation, the Google team proposed the BN [47] method to help train the network better. Average μ and variance σ^2 of elements of the BN layer in mini-batch training are used to normalize elements into a fixed range, subjected to natural distribution. Considering the effect of the normalization on performance, the authors implemented two learnable parameters, γ, and β, to scale and shift values, respectively.

Because of the LHE scheme, the inference latency of the PPDL model is highly related to the predefined level. Deploying the BN layer increases the MD, which sets a higher level and leads to longer latency. Since the BN layer is adopted into the typical NN model and then transformed for the NN model for ciphertext inference, its parameters μ, variance σ^2, scale γ, and shift β are deployed with multiplication. These four parameters are implemented in several arithmetic operations, which increases the MD. The CM is a required tool to reduce the MD. It focuses on coalescing parameters used in the BN layer with the other layers, such as the activation layer. Ioffe [47] et al. deployed the BN layer after each activation layer to improve accuracy. We point out two schemes of fusing the BN and activation layer in the encrypted model: one is Convolution-Activation-BN, which we call the CAB scheme, and the other is Convolution-BN-Activation, which we call the CBA scheme. We develop the fused layer via the CM as the mapping layer, described in detail in Section 3.4.2.

Pack Layer

The ciphertexts are the chaotic vectors that must be rearranged before feeding feature maps into the Fully Connected (FC) layer. Due to the vector-matrix multiplication in the FC layer, the model needs to pack these ciphertexts into one ciphertext whose valid slots must be equal to the input size of the following FC layer; otherwise, the model is unable to perform the HE addition. The required method is flattening the valid slots of all ciphertexts into one ciphertext. However, there are two problems with this method:

I Ciphertext has redundant slots that do not casually concatenate the rear with the head.

II Each ciphertext is chaotic, which means that the valid slots of the ciphertext are not neighbors.

The pack layer concatenates all the valid slots located along the length of the flattened channel vector. The length of slots of one ciphertext is fixed, whose size equals $N/2$, which is half of the polynomial degree N. It is not feasible to connect the end of one ciphertext to the head of another ciphertext, as it would exceed the maximum length. Thus, concatenating slots equal to the flattened channel's length rather than the ciphertext length would be a better alternative. However, there is a problem. If the length or the number of ciphertexts is too large, the concatenated slots will exceed the specified length, which implies that the valid slots are sparse. So, a well-structured output vector is needed in the FC layer. We propose the algorithm Flatten Multi-ciphertext to Single-ciphertext (FM2S) to pack ciphertexts into a single ciphertext, written down as Algorithm 7.

Fully Connected Layer

As for the FC layer of the NN model, the filter is a matrix of size, which shapes as an "input channel $\times$ output channel," and involves matrix multiplication of the input and the filter matrix. In the privacy-preserving CNN, before feeding feature maps into the FC layer, feature maps should be flattened into one ciphertext in the mentioned pack layer, which shapes a single vector. The input and the output of the FC layer are presented in a single ciphertext. However, the ciphertext encrypted by the HE does not support matrix multiplication. Consequently, we estimated using the HE rotation and HE multiplication to execute the matrix multiplication function. We change the matrix-vector multiplication scheme into a vector-matrix multiplication scheme. We leverage the approach proposed by Halevi and Shoup [80], a multiplication operation between vector, tensor or matrix.

The first phase of this layer is to pre-process the weight matrix. Since the output channel is smaller than the input channel, it is crucial to pad the matrix by zero to the square matrix of size "input channel $\times$ input channel." Next, the padded matrix is diagonalized along with the raw. Vector-matrix multiplication performs element-wise multiplication between the ciphertext and plaintext weight matrix, which can

Algorithm 7. Flatten Multi-ciphertexts to Single-ciphertext (FM2S)

Input: input feature map $[x]$, last output feature map size n, valid list v, mask m

Output: packed ciphertext $[x\_pac]$

Obtain channels of $[x] : c \longleftarrow len([x])$
for $i = 1, ..., n^2 - 1$ **do**
    $rot\_idx_i \longleftarrow v_i - i$
end for
for $j = 0, ..., c - 1$ **do**
    $[y] \leftarrow [x_j]$
    $mask \leftarrow Encode(m)$
    $[Z_0] \leftarrow HEmult([y], mask)$
    for $k = 0, ..., n^2 - 1$ **do**
        $mask_{v_{k+1}} \leftarrow 1$
        $[Z_{k+1}] \leftarrow Rot(HEmult([y], mask), rot\_idx_k)$
    end for
    $[y] \leftarrow HEadd([Z_0], ..., [Z_{n^2-1}])$
    $[Y_i] \leftarrow Rot([y], -(n \times n \times i))$
end for
$[x\_pac] \leftarrow HEadd([Y_0], ..., [Y_{c-1}])$
return $[x\_pac]$

only be achieved by the HE rotation.

We found that the vector-matrix multiplication has an unnecessary step, which increases latency. Assuming the input channel is I, and the output channel is O, it needs $2(I - 1)$ times the HE rotations and one time the HE addition, where there is $2(I - 1)/2$ times the HE rotation to the left and $2(I - 1)/2$ times the HE rotation to the right linearly. To reduce the HE operation, we propose the method that implements fewer than $2(I - 1)/2$ times the HE rotation to the right linearly, as shown in Algorithm 8. Before reviewing the actual mechanism of the proposed algorithm, Figure 3.6 should be checked for several operations with an easy example. Thus, the instance of the FC layer is presented in Figure 3.7.

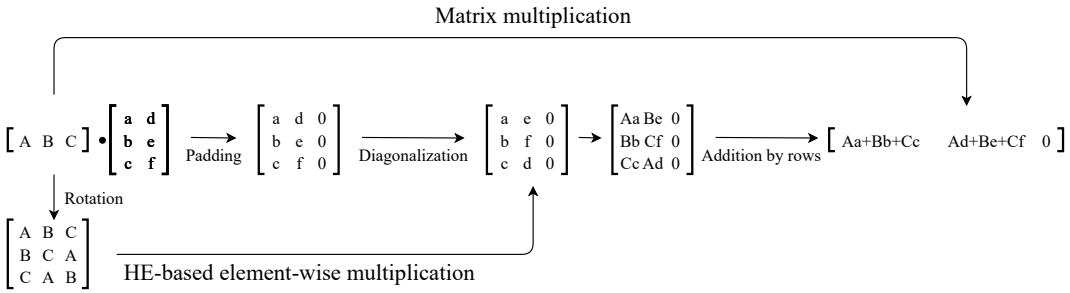


Figure 3.6. The easy example to show how to compute the HE matrix multiplication between a 1×3 ciphertext and a 3×1 weight without bias, which outputs a 1×3 ciphertext.

In Figure 3.7, elements of the part η are computed as zero in the output, which implies that the elements do not affect the output. Currently, as the HE rotation is a linear operation, intersection $I_{\delta,\eta}$ of the part δ and part η are in serial, which means that the HE rotation can assemble linear transformation simultaneously. On the other hand, the intersection $I_{\zeta,\eta}$ of the part ζ and η are not in serial, which means the transformation is non-linear, and two times of the HE rotations are necessary. The previous method costs $2(I-1)$ for the HE rotations in the FC layer. As for the proposed enhanced approach, the intersection $I_{\delta,\eta}$ requires $I-O$ times the HE rotations, which runs one time for the HE rotation to each line; the intersection $I_{\zeta,\eta}$ expends $2(I-(I-O)-1) = 2(O-1)$ times for the HE rotations, which executes two for the HE rotations to each line. The number of the HE rotations is changed to $I+O-2$, i.e., the latency of this step has become $I+O-2/2(I-1)$ of the original. Thus, for the example mentioned above, which demonstrated that it is possible to ignore a part of the HE rotation moving to the right, the latency becomes $\frac{7+3-2}{2(7-1)} = \frac{2}{3}$ of the original.

The relationship between the input channel and the output channel is drawn in Figure 3.8. The $x-$ and $y-$ axes represent the FC layer's input and output channels. The $z-$axis is the ratio (%) of the number of the HE operations of original and improved methods. Before implementing the enhanced algorithm, the HE operation of the FC layer needs to be performed $2(I-1)$ times. After that, the number of the

HE operations becomes $I + O - 2$ times. This ratio implies that the latency of the FC layer can be reduced to somewhat the original by using an improved algorithm, depending on the number of input and output channels. A lower ratio means better results for the improved algorithm.

Algorithm 8. Vector-matrix Multiplication (VMM)

Input: input feature map $[x]$, weight matrix w, bias b

Output: ciphertext $[x']$

$[t] \leftarrow [x]$
Obtain input feature map size $m \leftarrow len(w)$
Obtain output feature map size $n \leftarrow len(b)$
Set boundary $e \leftarrow m - n + 1$
$w \leftarrow Diagonalization(Padding(w))$
for $i = 0, ..., m - 1$ **do**
    $w_i \leftarrow Encode(w_i)$
    $[Y_i] \leftarrow HEmult([t], w_i)$
    IF $i \neq 0$ *or* $i < e$:
        $[t] \leftarrow Rot([x], i + 1)$
    IF $i \neq m - 1$ *and* $i \geq e$:
        $[t] \leftarrow HEadd(Rot([x], i + 1), Rot([x], i + 1 - m))$
end for
$[Y] \leftarrow HEadd([Y_0], ..., [Y_{m-1}])$
$[Y] \leftarrow Rescale([Y])$
$B \leftarrow Encode(b)$
$[x'] \leftarrow HEadd([Y], B)$
return $[x']$

3.4.2 Batch Normalization with Coefficient Merging

To improve the accuracy and reduce the MD of ciphertext inference in the CNN, we process the BN layer with the CM. By employing qualitative modes of inquiry, we

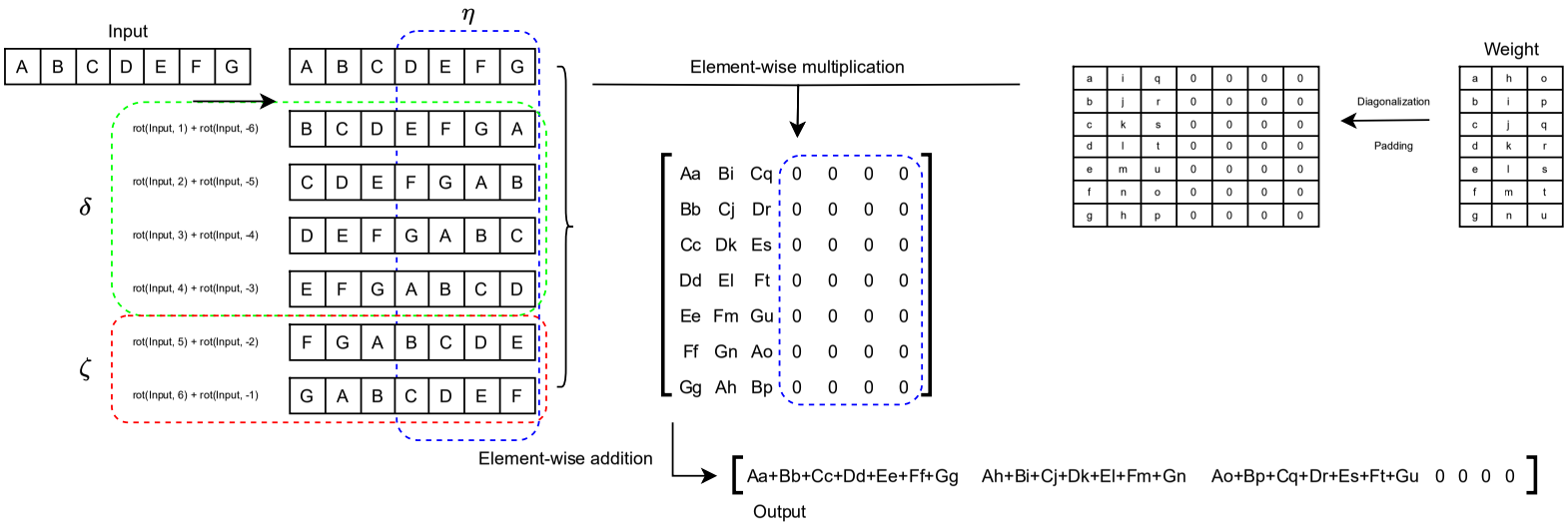


Figure 3.7. Vector-matrix multiplication. Assume the input is ciphertext with the first seven valid elements $[A \quad B \quad C \quad D \quad E \quad F \quad G]$, which has a shape of a 1×7 vector, and the filter is 7×3 matrix which implies that the input channel is seven and the output channel is three of the FC layer.

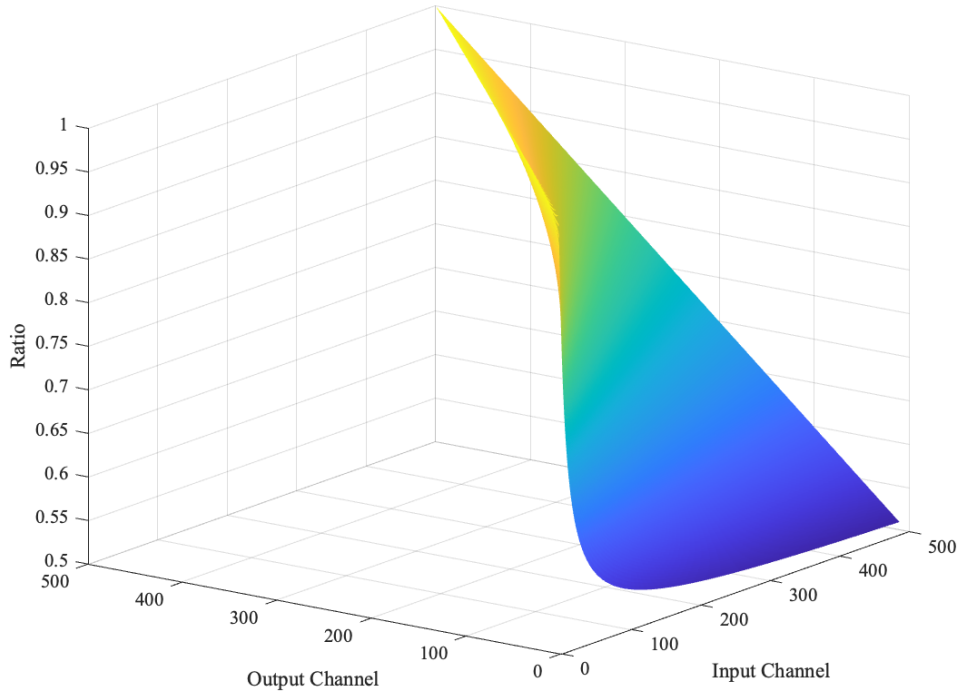


Figure 3.8. The relationship between the input channel and output channel of the FC layer.

attempt to illuminate the two different schemes in mathematics. The derivations of mathematical formulae of the schemes are shown as the following.

The formulae of the two schemes are described in order. A convolution layer can be written down as in Equation 3.2, where $[I_f]$ is the input feature map, $f_{11}, \ldots, f_{1f}, \ldots, f_{ff}$ are weight elements of the filter of size $f \times f$, and $[X_{conv}]$ is the output of *rotation & accumulation*. Furthermore, $[Y_{conv}]$ can be obtained by the *mask computation*, which leverages a mask m to remove the redundant slots "#," as in Equation 3.3. The process of obtaining $[X_{conv}]$ and $[Y_{conv}]$ are shown in Figure 3.4.

$$[X_{conv}] = [I_f]f_{11} + \cdots + rot([I_f], f \times f - 1)f_{ff} \tag{3.2}$$

$$[Y_{conv}] = m[X_{conv}] \tag{3.3}$$

Convolution-Activation-BN (CAB) scheme: A second-order polynomial is used to approximate the activation function, as in Equation 3.4. In the BN layer, the normalized $[\overline{Y_{act}}]$ is calculated by the input $[Y_{act}]$, the mean μ and variance σ^2. In addition, due to the gradient dispersion problem, the output is multiplied by the scale γ or weight, and the shift β or bias is also added. After simplifying the formula, the actual second-order polynomial approximation is expressed as in Equation 3.5. We replace the activation function with the square function, which implies that the coefficients a, b, and c of the second-order function are one, zero, and zero, respectively. The mask m used in the convolution layer is a vector containing only zero and one. Its objective is to obtain a result without the redundant slots "#." So the value of m is regarded as one logically, i.e., the value of m^2 is viewed as one. In conclusion, for the specific example of a square function, the output feature map $[Y_{bn}]$ of the CAB scheme is changed into $[Y'_{bn}]$, as in Equation 3.6. After reducing the MD, as proposed by Ishiyama et al. [9], the output of the CAB scheme is formulated as in Equation 3.7.

$$[Y_{act}] = a[Y^2_{conv}] + b[Y_{conv}] + c \tag{3.4}$$

$$\begin{aligned}[Y_{bn}] &= \gamma[\overline{Y_{act}}] + \beta \\ &= \frac{\gamma a m^2}{\sqrt{\sigma^2 + \epsilon}}[X^2_{conv}] + \frac{\gamma b m}{\sqrt{\sigma^2 + \epsilon}}[X_{conv}] \\ &\quad + (\beta - \frac{\gamma(\mu - c)}{\sqrt{\sigma^2 + \epsilon}})\end{aligned} \tag{3.5}$$

$$[Y'_{bn}] = \frac{\gamma}{\sqrt{\sigma^2+\epsilon}}[X^2_{conv}] + (\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}) \tag{3.6}$$

$$= [X^2_{conv}] + B \tag{3.7}$$

$$where \quad B = (\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})/\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}$$

Convolution-BN-Activation (CBA) scheme: In the BN layer, the normalized $[\overline{Y_{conv}}]$ is calculated by the mean μ and variance σ^2 with the input $[Y_{conv}]$. In addition, $[\overline{Y_{conv}}]$ also needs to be multiplied by the scale γ and added with the shift β; the resulting well-structured formula is described in Equation 3.8. For the CBA scheme, the output feature map of the BN layer is followed by the activation layer, as in Equation 3.10. Furthermore, Equation 3.10 substituted $[Y_{conv}]$ with $[X_{conv}]$ combined within Equation 3.2. As for the specific example of a square function, the output feature map $[Y_{act}]$ of the CBA scheme is changed into $[Y'_{act}]$ after the CM, as in Equation 3.11. After reducing the MD, as proposed by [9], the output of the CBA scheme is formulated as in Equation 3.12.

$$[Y_{bn}] = \gamma[\overline{Y_{conv}}] + \beta \tag{3.8}$$

$$[Y_{act}] = a[Y^2_{bn}] + b[Y_{bn}] + c \tag{3.9}$$

$$= \frac{a\gamma^2 m^2}{\sigma^2+\epsilon}[X^2_{conv}] + [2a(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}$$
$$+ \frac{b\gamma}{\sqrt{\sigma^2+\epsilon}}]m[X_{conv}] + [a(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})^2$$
$$+ b(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}) + c] \tag{3.10}$$

$$[Y'_{act}] = \frac{\gamma^2}{\sigma^2+\epsilon}[X_{conv}]^2 + [2(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}) \frac{\gamma}{\sqrt{\sigma^2+\epsilon}}][X_{conv}] + (\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})^2 \tag{3.11}$$

$$= [X^2_{conv}] + C[X_{conv}] + D \tag{3.12}$$

$$where \quad C = [2(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}]/\frac{\gamma^2}{\sigma^2+\epsilon}$$

$$D = (\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})^2/\frac{\gamma^2}{\sigma^2+\epsilon}$$

The MD of the mapping layer becomes one after the CM and optimization. The BN and activation layers are separate when the NN model is deployed over plaintext. Still, they are fused into one, the mapping layer, when deployed over ciphertext, to optimize the MD. The mapping layer produces completely different outputs for the two schemes when applied to the output of the convolution layer. The CAB scheme produced a second-degree term with one constant, while the CBA scheme produced an additional first-degree term. These different mappings result in their disparate accuracy and latency.

3.5 Experiment

To verify if the latency of encryption and decryption for the CHE is lower than that for the PiHE, we compare their performances for one instance. Furthermore, we apply the CHE to the five previous NN architectures [6] [7] [8] [9] [11] and measured the accuracy and latency of ciphertext inference for comparison.

3.5.1 Datasets and Networks

Datasets. We adopt the MNIST and CIFAR-10 datasets to evaluate our proposed methods. We use the MNIST and CIFAR-10 datasets as the same datasets used in previous papers. Both datasets are standard datasets in this research area. The MNIST is a dataset of handwritten digits with a training set of 60, 000 images and a

Table 3.3. The network architectures of privacy-preserving neural networks. C, A, P, B, and F denote convolution, activation, pooling, batch normalization, and fully connected layers, respectively.

Network	Architecture
CryptoNets [6]	C-A-P-A-F
Light CNN [7]	C-P-C-P-F-B-A-F
HCNN [11]	C-A-C-A-F
CNN A [8]	C-P-C-A-P-F-F
CNN B [9]	C-B-A-C-B-A-F

test set of $10,000$ images, each of size 28×28, divided into 10 classes of numbers from zero to nine. It is a gray image dataset, where each image has a single channel. The CIFAR-10 dataset consists of $60,000$ color images, each of the size 32×32 in 10 classes, with $6,000$ images per class. There are $50,000$ in training images and $10,000$ in test images. Unlike the former, each image in CIFAR-10 has three channels.

Networks. We adopt the CNN to evaluate performance and compare the five networks shown in Table 3.3 with the proposed CHE. Initially, these studies adopted the pixel-wise approach. We replicate their architecture precisely and apply the CHE and the CM schemes to improve the models' performance.

3.5.2 Experimental Setup

We run all comparison models on the same Linux server in parallel, equipped with Intel Xeon E5-2660 with 126GB memory and 40 cores. We only focus on the ciphertext inference stage and use Pytorch [81] to train all the NN models with plaintext. The epoch is set as 100, and the Adaptive moment estimation (Adam) [46] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ as the optimization, the learning rate as 0.01. We adopt the Microsoft Simple Encrypted Arithmetic Library (SEAL) 3.7 [82] in Python 3.9 [83] using Python-SEAL [84] to encrypt and decrypt data with the RNS-CKKS

HE scheme [64]. All the PPDL models in our experiments could achieve the 128−bit security level.

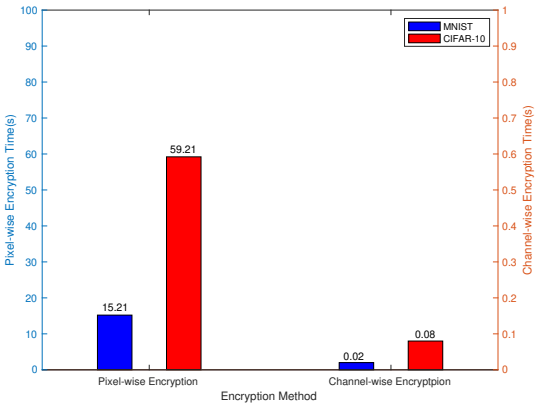
To simulate an actual situation when the user sends one query to the server, we set the inference image of every epoch to be one. Furthermore, the HE flattens each image channel into one vector and encrypts it into ciphertext. For example, each image of the CIFAR-10 has three channels flattened into three vectors and encrypted as three ciphertexts. So, the model gets a list with three ciphertexts as the input to the HE-friendly CNN. After running over the model, the model's output is a single ciphertext with valuable front parts with 10 slots and redundant rear parts or other slots. To obtain the prediction, the user cuts down the decrypted output and feeds it into the SoftMax function [45]. The convolution, pack, and FC layers are the three tough layers in the model used for parallel computing; serial calculation is used for the rest.

3.5.3 Result

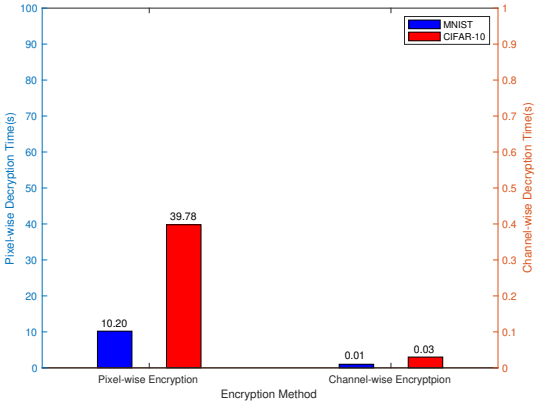
Encryption and Decryption

We first evaluate the time required for encryption and decryption in the CHE and PiHE simulating end-to-end. This experiment encrypts the same instance into ciphertext using the CHE and PiHE, respectively, and then decrypts ciphertext into plaintext, counting the encryption and decryption time. The encryption and decryption run time depends on the number of ciphertexts.

The aim is to validate if the former takes less encryption and decryption time than the latter. A random instance is chosen from the MNIST and CIFAR-10, and the example image is processed using the two approaches while logging the encryption and decryption times. The test is performed 100 times to obtain an average result, shown in Figure 3.9. Statistically, under datasets, the MNIST and CIFAR-10, deploying with the CHE, costs 0.02 seconds and 0.08 seconds for encryption and costs 0.01 seconds and 0.03 seconds for decryption, respectively. On the contrary, the PiHE takes 15.21 seconds and 59.21 seconds for encryption and 10.20 seconds and 39.78 seconds for decryption, respectively. As expected, the CHE takes a significantly shorter time for encryption and decryption.



(a). Encryption Time of Example in the CHE



(b). Decryption Time of Example in the CHE

Figure 3.9. Encryption and decryption latency on the datasets MNIST and CIFAR-10 datasets for one instance.

Coefficients Merging Schemes

To test the effect of different schemes on the performance, a simple model HCNN [11] and the MNIST dataset are chosen. As shown in Table 3.4, the CAB and CBA schemes obtained the accuracy of 98.50% and 99.03%, respectively, for ciphertext inference. The attained latency is 10.35 seconds and 10.96 seconds, respectively. The experimental results reveal that the CBA scheme has higher accuracy and longer latency, which conforms with the formulae we derived in Section 3.4.2.

Table 3.4. The accuracy and latency of different schemes in the CHE

Dataset	Model	Scheme	Ciphertext Inference Accuracy (%)	Inference Latency (second)
MNIST	HCNN	CAB	**98.50**	**10.35**
		CBA	**99.03**	**10.96**

Due to the fused layer output, the results of the CBA scheme are stable with high accuracy, but the latency becomes a bit long. In contrast, as the CAB scheme does not have enough lower-order terms to analyze the details of the feature map, certain elements are ignored during inference. The CAB scheme results in a shorter latency but reduced accuracy.

Accuracy and Latency

Accuracy is the primary performance measuring a parameter of the inference. As the CBA scheme achieves higher accuracy than the CAB scheme, thus, we perform the CHE with the CBA scheme to verify whether the CHE implementation improves the accuracy and latency on the MNIST and CIFAR-10. The ciphertext inference time is logged, and all recorded values are the average of 10 times experiments.

Table 3.5 shows the results of the five models and compares the accuracy and latency of the original models and the proposed model on the MNIST. The proposed method leverages the CHE to process the whole model, and the fused layer employs the CBA scheme. It achieves an accuracy of 99.00% for the CryptoNets [6], 98.05% for the light CNN [7], 99.30% for the HCNN [11], 99.32% for the CNN A [8], and 99.30% for the CNN B [9]. The corresponding latency values for the five models were 15.22 seconds, 17.89 seconds, 10.96 seconds, 16.14 seconds, and 7.76 seconds, respectively. Furthermore, the speedups are 37.45, – "no previous inference latency," 1.29, 23.96, and 2.73, respectively.

The results show better performance than the previous five works, especially speedup. As only the work of the CNN B evaluated the CIFAR-10 dataset, Table 3.6 presents the single result of the CNN B model and compares it with the accuracy and latency of the original and proposed models on the CIFAR-10. Our method

Table 3.5. The inference latency and accuracy of previous models on the MNIST. A multi-processing situation evaluates latency. The L is a level of the model, the Scale is a scale factor of the HE, the N is a polynomial degree, and the Q is a modulo.

Network	L	Scale (b)	N (b)	Q (b)	PiHE Inference Accuracy in paper (%)	**CHE Proposed Method Accuracy (%)**	PiHE Inference Time in paper (second)	**CHE Proposed Method Inference Time (second)**	**Speedup**
CryptoNets	7	2^{30}	16384	330	98.95 [6]	**99.00 (+0.05)**	570.00 [6]	**15.22 (−554.78)**	**37.45**
light CNN	10	2^{30}	16384	420	97.91 [7]	**98.05 (+0.14)**	-	**17.89 (-)**	**-**
HCNN	8	2^{30}	16384	360	99.00 [11]	**99.30 (+0.30)**	14.11 [11]	**10.96 (−3.15)**	**1.29**
CNN A	10	2^{30}	16384	420	99.25 [8]	**99.32 (+0.07)**	386.70 [8]	**16.14 (−370.56)**	**23.96**
CNN B	8	2^{30}	16384	360	99.18 [9]	**99.30 (+0.12)**	21.15 [9]	**7.76 (−13.39)**	**2.73**

Table 3.6. The inference latency and accuracy of previous model CNN B on the CIFAR-10. The L is a level of the model, the Scale is a scale factor of the HE, the N is a polynomial degree, and the Q is a modulo.

Network	L	Scale (b)	N (b)	Q (b)	PiHE Inference Accuracy in paper (%)	**CHE Proposed Method Accuracy** (%)	PiHE Inference Time in paper (second)	**CHE Proposed Method Inference Time (second)**	**Speedup**
CNN B	8	2^{30}	16384	360	76.37 [9]	**76.40 (+0.03)**	1038.00 [9]	**111.91 (−926.09)**	**9.28**

leverages the CHE to process all models, and the BN layer employs the CBA scheme. The proposed model achieves an accuracy of 76.40%, latency of 111.91 seconds, and speedup of 9.28. Thus, in other words, the proposed model establishes better results for this case.

3.6 Discussion

To perform inference on large datasets in the NN model, researchers packed the same pixel of multiple images in a single ciphertext, which could test many ciphertext images in one inference, called the PiHE. Previous research [8] [11] [69] have shown reasonable accuracy and latency in interactive and non-interactive paradigms.

The PiHE uses tensors to process data, and each element of a tensor is ciphertext. The CHE leverages data upon ciphertext; the data is ciphertext rather than plaintext. Dathathri et al. [57] and Lou et al. [58] implemented a similar CHE presented in their figures. However, it is challenging to derive their exact algorithm from the figures in which it has been presented. Also, their method is used only for the convolution layer. At the same time, our proposed algorithm applies to all kinds of layers in the NN, including the convolution, pooling, activation, BN, and FC layers.

Consistent with previous studies, the work of this chapter shows that the CHE can achieve higher accuracy and shorter latency than current PPDL models under multi-processing. Applying the CHE to encrypt data into ciphertext in the number of channels, we find out that the proposed HE-friendly algorithms for the CHE in the CNN are prudent and reliable. Although the PiHE can package a large number of pixels together, in the actual inference process, pixels in the same channel are independent; that is, they are not in the same ciphertext. In contrast, the advantage of the CHE is that during the actual inference process, pixels in the same channel are all packaged in the same ciphertext, leading to performance differences. The algorithm is predominantly expressed in different layers and easily exploited to transform the NN models. As for the generalization of the CHE, since the algorithm inside the model has been adapted to the ciphertext calculation, it is possible to transform any NN model using the CHE. At the same time, it is only necessary to transform the

input data into vector data encrypted by the HE for ciphertext inference.

The CAB and CBA schemes show only minor significant differences in accuracy and latency. It must be pointed out that, due to the CAB formula, only the profile of the image is used for the results, lowering the scheme's accuracy. On the contrary, due to the CBA formula, the image profile and other details are used for the results, leading to a higher frequency but longer latency. Because there are different output polynomial terms in the fused layer, the CAB scheme is more suitable for quick inference over ciphertext, where the focus is on central architecture, and the CBA scheme is more suitable for precise inference over ciphertext to not only see the main body but also pay attention to the details. We believe that the minor differences in the accuracy and latency are also because of the implementation of the second-order square function. Nevertheless, the final formulae are not hugely different after deploying the CM through the proposed CAB and CBA schemes because the CBA has only one additional first-order term using the square function. If higher-order polynomials are deployed, the final formulae after the CM present more significant differences, increasing the variation in the accuracy and latency. Technically, the CBA scheme provides a more stable analysis in ciphertext inference, and the CAB scheme provides a quicker analysis.

Despite the CHE's preliminary character, this chapter's study indicates that reasoning and corroborating the fundamentals of data structures can prove its effectiveness and efficiency. However, there is a restriction on inference latency due to the sophistication and black-box nature of ciphertext. In addition, the differences in the accuracies and latencies of the CBA and CAB could be significant if a function more sophisticated than the square function is used. Since there is no source code to reproduce the previous works, we leave the direct comparison with the earlier studies under a specific platform as a future study. Furthermore, the most significant limitation of this work is the HE method. The ultimate goal of the HE-based method is to operate using Fully HE (FHE). However, due to computational limitations and differences in physical operations, researchers currently opt for a compromise approach of using the LHE for operations to make methods based on the HE applicable in practice. While this method significantly reduces computational overhead,

professionals must pre-calculate the MD to set as a threshold. If this threshold is exceeded, the result on the ciphertext will not be correctly restored due to excessive noise.

Future iterations of inference over ciphertext from the CHE to instance-wise encryption, which encrypts one instance as the single ciphertext, may demonstrate even greater potency. Hopefully, the experimental results improve inference over ciphertext, significantly changing the basic data structures used for the PPDL inference.

3.7 Conclusion

The study of this chapter sets out to improve the performance of the HE-based PPDL by combining the proposed approaches of the CHE and the BN layer with the CM. It also discusses several related algorithms in detail and the computation scheme "Onion." The CHE implements ciphertext inference for the end-user in the non-interactive paradigm. The BN layer with the CM improves the accuracy and decreases the latency by reducing the MD using two proposed schemes, the CBA and CAB. The proposed method achieves the highest accuracy of 99.32% and the shortest latency of 7.76 seconds on the MNIST dataset compared to five previous architectures. It also attains an accuracy of 76.4% and a latency of 111.91 seconds on the CIFAR-10. Thus, our experiments demonstrate that the CHE can serve as a tool to design a more robust and flexible PPDL model that performs ciphertext inference in the CNN with better accuracy and latency. In future work, we will target more challenging problems with processing on graphical processing units, actual datasets, and deeper NN. We aim to achieve lower latency and higher accuracy for instance-wise ciphertext inference that encrypts one instance as the single ciphertext.

Chapter 4

Contribution 3: The Trade-offs of Privacy, Utility, and Efficiency in Differential Privacy-Enabled VQ-VAE for Image Generation *[1]

4.1 Introduction

Deep Learning (DL) models have brought convenience to the public's lives. However, behind this convenience lies reliance on vast amounts of data, especially various sensitive data closely related to individuals. Therefore, when designing algorithms, privacy-preserving of data needs to be considered in the DL.

*[1] This chapter is based on the paper: [Towards Optimal Privacy-Preserving: The Trade-offs of Privacy, Utility, and Efficiency in Differential Privacy-Enabled VQ-VAE for Image Generation (in application)].

Researchers refer to this as "Privacy Computing," [85] a concept that allows for the circulation and deep exploration of data value while ensuring the security of the rights and interests of data owners and safeguarding personal privacy. The work by Torkzadehmahani et al. [19] effectively integrated the concept of Differential Privacy (DP) [74] with Generative Adversarial Network (GAN) [86] and proposed a novel method for satisfying the DP constraints in the GAN framework by modifying the training procedure and architecture of the GAN. Contrary to generating images with the DP in a post-processing stage, they applied the DP during the learning process itself, aiming to generate synthetic data that maintains the privacy of individuals in the original dataset [20]. Jiang et al. [21] proposed a novel training method that incorporated DP into pre-trained Variational AutoEncoder (VAE) [87]. By integrating the DP into the pre-training phase, their approach aims to ensure that the privacy of individuals in the original dataset is respected while preserving the utility of the generated synthetic data. While effective in the VAE context, this approach's pioneering efforts in introducing the DP into the training process of the VAE offer valuable insights. Furthermore, the Vector Quantized-VAE (VQ-VAE) [88] is a state-of-the-art (SOTA) generative model known for its capacity to learn discrete representations and synthesize high-quality images compared with the VAE. However, the standard VQ-VAE framework does not inherently provide any privacy guarantees.

Consequently, the prospect of adapting the VQ-VAE to incorporate the DP presents an exciting opportunity to propel advancements in the field of privacy-preserving image reconstruction and generation. To ensure the DP, noise is often added during the learning process, particularly during gradient computation, which is crucial in algorithms like Stochastic Gradient Descent (SGD) [89]. This method can help prevent the model from memorizing specific data instances, thereby protecting individual data privacy. Furthermore, when applying the DP, one needs to consider the privacy budget ϵ [74]. As the privacy budget decreases, more noise is added, leading to higher privacy-preserving but lower accuracy. Directly applying the DP and adjusting the privacy budget for the VQ-VAE model can impact the final outcome. To reduce the privacy budget as much as possible without compromising

accuracy, the work of this chapter introduces a new training method for the VQ-VAE with the DP. Our method builds upon the existing VQ-VAE framework by incorporating the Gaussian-DP mechanism [90] with implementing DP-SGD with the Tensorflow Privacy library to ensure privacy-preserving for sensitive image data. The critical contributions of our work are as follows:

- We present a comprehensive overview of the approach that introduces new possibilities for secure data sharing, as synthetic datasets produced by a privacy-preserving generative model could be distributed freely without infringing on privacy regulations or ethical guidelines.
- We introduce a privacy-preserving VQ-VAE architecture that incorporates the DP mechanisms, enabling the reconstruction and generation of realistic images while protecting the privacy of the underlying data and conserving the privacy budget ϵ.
- We deliver an in-depth analysis of the trade-offs between privacy, utility, and computational efficiency in our proposed model. This analysis illuminates the key factors that influence the performance of our privacy-preserving VQ-VAE within the context of private computing.

4.2 Preliminary

In this section, we offer an overview of the core concepts and techniques that underpin our work. We briefly introduce the concept of the DP and discuss the generative model, primarily focusing on VQ-VAE.

4.2.1 Differential Privacy

One rigorously defined privacy tool is the DP, which is utilized to publish statistics across a wide range of domains and applications. Specifically, deep generative models have been proposed for differential private implementation, aiming to preserve data privacy while generating synthetic data. The DP has seen successful application across various data analysis tasks, but its adoption within the realm of image

generation remains limited. The DP is a formal framework designed to protect the privacy of individual data points in a dataset while still allowing for aggregate statistical analysis. The DP achieves this by introducing a controlled amount of noise to either the data or model parameters, ensuring that the presence or absence of a single data point does not significantly influence the result of any analysis or computation. The privacy guarantee the DP provides is encapsulated by the privacy budget ϵ. A smaller value of ϵ corresponds to a stronger privacy guarantee.

$$Pr[M(x) \in S] \leq exp(\epsilon)Pr[M(y) \in S] + \delta \tag{4.1}$$

In Equation 4.1, the M denotes a randomized algorithm, while the S represents all potential output of the M that could be predicted. The variable x stands for entries in the database, whereas the y signifies entries in the parallel database. The symbol ϵ indicates the maximum distance between a query on database x and the same query on database y. Finally, the δ implies a probability of information accidentally being leaked.

4.2.2 Generative Model

Generative models [91] constitute a class of Machine Learning (ML) models designed to learn the underlying probability distribution of a dataset. This allows for the creation of new samples that mimic the original data. The three widely known generative models are the GAN, VAE, and Auto-Regressive (AR) models [92]. Generally, the aim of a generative model is to maximize the likelihood, which involves two different schemes: implicit density and explicit density [93]. These are used to estimate the probability distribution, which has been shown in Equation 4.2.

$$\theta^* = \underset{\theta}{\text{argmax}} \mathbb{E}_{x \sim \text{data}} \log p_{model}(x \mid \theta) \tag{4.2}$$

4.2.3 Vector Quantized-Variational AutoEncoder Model

The VQ-VAE is an extension of the traditional VAE framework that uses a discrete latent space, also known as the codebook, instead of a continuous layer. In the VQ-VAE setup, an encoder network produces continuous latent representations, which are then quantized to a discrete set of codebook vectors. This effectively allows the model to learn a discrete latent space. A decoder network then uses the quantized latent codes as input to generate the output samples. The employment of discrete latent representations enables the VQ-VAE model to generate higher-quality images while maintaining enhanced control over the generated content. The operation of this model can be readily understood by examining the loss function presented in Equation 4.3.

$$L = \log p(x \mid z_q(x)) + \|sg[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - sg[e]\|_2^2 \tag{4.3}$$

During the training phase of the VQ-VAE, the original data is passed through an encoder to a continuous latent space, which is then discretized using a codebook of vectors and is regarded as the vector quantization step. The model is trained to reconstruct the original data from the codebook indices, which results in a learned mapping from the discrete latent to the data. After training, the categorical prior is replaced with an AR model, which learns the dependencies between the latent codes. This model, called PixelCNN [94], an AR model, treats each code as dependent on the previous ones, allowing it to model complex joint distributions over sequences of latent codes.

4.2.4 Threat Model

In the context of the DP and VQ-VAE, the threat model typically addresses the following concerns; the detail is shown in Figure 4.1:

I Reconstruction attacks: an attacker tries to reconstruct original input data from the model's outputs or parameters. The threat is that if the model

remembers too much about specific training data, it might leak sensitive information when queried.

II Membership inference attacks: an attacker attempts to determine whether a particular data was used in the model's training dataset. This can be particularly concerning if the presence of data in the training data is sensitive.

III Model inversion attacks: similar to reconstruction, but here, the attacker focuses on inferring sensitive features or inputs from the model's predictions.

They are printed as concerns I and II since after obtaining a well-trained VQ-VAE model, only the decoder and PixelCNN of the VQ-VAE have been released to the user, which allows the user to generate synthetic data which has no secure or privacy concerns. What's more, implementing the DP to add noise into the model while training gives a formal guarantee that individual-level information about participants in the database is not leaked, which aims to concern II. At the same time, due to the presence of concern III, the published model also contains privacy related to the model itself, namely the model's parameters, such as weight and bias. When users utilize the VQ-VAE model, they are not only generating synthetic data but also using the reconstruction function to obtain de-privatized data. In this feature, users have the opportunity to access the complete VQ-VAE model, thereby potentially launching attacks. Therefore, we have also employed the DP-SGD training method to protect the parameters of the model from being leaked, which also serves to safeguard against threats related to concern III.

4.3 Related Work

In this section, we review the literature relevant to our work, focusing on three main areas: generative models, the VQ-VAE, and Privacy-preserving DL (PPDL) [95] with an emphasis on the DP.

Generative models have been a popular area of research in recent years, with numerous advancements in unsupervised and semi-supervised learning techniques. Some of the widely used generative models include the VAE [21] [87] [96] [97] [98], the GAN [86] [99], and the AR models such as PixelRNN [100]. These models

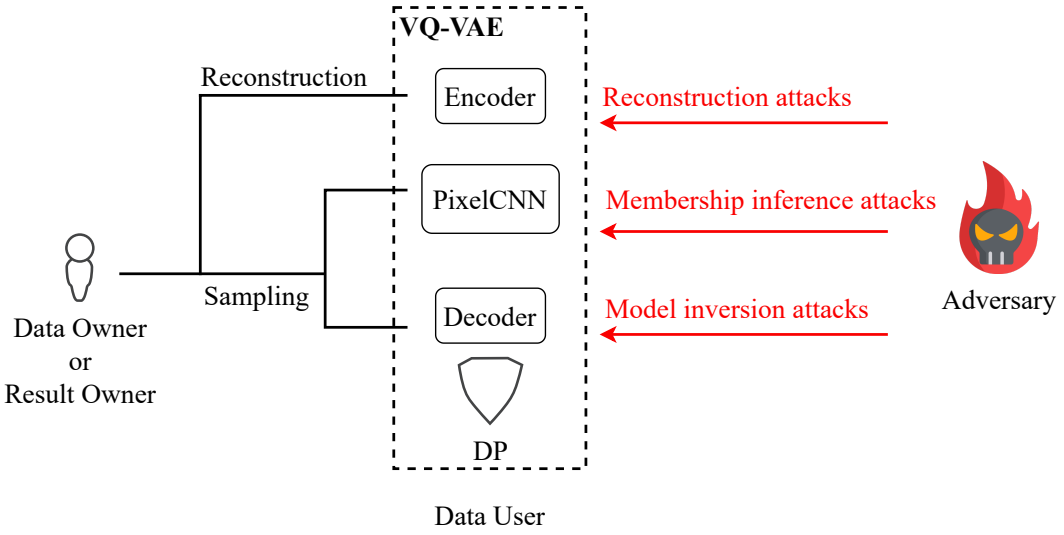


Figure 4.1. The threat model of the DP and VQ-VAE. There are three typical concerns: I Reconstruction attacks, II Membership inference attacks, and III Model inversion attacks.

have been employed to generate realistic images, learn powerful representations, and enable novel applications in domains like Computer Vision (CV), Natural Language Processing (NLP), and Reinforcement Learning (RL).

The VQ-VAE emerged as a significant advancement in the field of generative modeling, combining the strengths of the VAE and discrete latent representations. The VQ-VAE exploits a vector quantization layer to map continuous encodings to a discrete latent space, enabling better control over the generated images and improved sample quality. Since their introduction, the VQ-VAE has been further developed and extended, with hierarchical architectures [101] leading to even higher-quality image synthesis.

The importance of privacy-preserving in the DL, also named the PPDL, has grown significantly, with various techniques being proposed to protect sensitive data during model training and inference. One of the most prominent frameworks for privacy-preserving is the DP, which introduces controlled noise to data or model parameters to ensure that the presence or absence of any single data point does not

significantly affect the model's outputs. The DP has been applied to a wide range of the DL tasks, including supervised learning [102], unsupervised learning [44], and Federated Learning (FL) [17].

There have been several efforts to incorporate the DP into generative models, primarily focusing on the VAE and GAN [19] [103] [104]. These approaches generally involve modifying the model architecture, training procedure, or loss functions to satisfy the DP constraints [105]. However, the integration of the DP into the VQ-VAE has been relatively unexplored, leaving room for further investigation and development.

4.4 Methodology

In this section, we discuss the methodology of integrating the DP in the VQ-VAE model, emphasizing the privacy budget ϵ. Moreover, we follow the proof for applying the DP in the VQ-VAE provided by Jiang et al. [21]. We observe that directly embedding the DP in the VQ-VAE model for data processing results in using the original data twice an epoch during model training [101]. The DP quantifies privacy loss using a metric called the privacy budget, typically denoted by epsilon ϵ. Each query or computation on the dataset consumes a portion of this privacy budget. The more the data is used, which means the more queries or computations are performed, the more the privacy budget is expended.

The fact is that during the training, there are two times accessing training data each epoch. In the scenario of the VQ-VAE with the DP, every time the model is trained, updated, or a new query is made, a portion of the privacy budget is used up. As the cumulative usage of the data increases, so does the total amount of privacy budget spent, increasing the risk of privacy exposure. This is because each use of the data adds a little more "information leakage," gradually eroding the privacy guarantees.

To mitigate this issue, we proposed modifying the conventional training process for the DP integrating with the VQ-VAE model. This modification ensures that the original data is accessed only once an epoch during training, thus reducing the

privacy budget and increasing the level of data privacy protection without compromising performance. For simplicity, we denote the vanilla training flow abbreviated as the VTF, Algorothm 9 [101], and the revised training flow abbreviated as the RTF, Algorithm 10, which merges the separate gradient updates of the VQ-VAE and PixelCNN models into a single gradient update in each epoch.

4.4.1 Data Flow

Figure 4.2 illustrates the detailed data flow in the DP-integrated VQ-VAE model. This model's data flow comprises two distinct training flows: the VQ-VAE model, which is represented by a black line, and the PixelCNN model, which is represented by a blue line, constitute the complete VQ-VAE model. The black line, referred to as "reconstruction," depicts the VQ-VAE model's training flow. Upon completion of the VQ-VAE model training, the yellow line illustrates the data flow for generating synthetic data from the latent distribution using randomized noise, known as "sampling or generation." A naive implementation of DP in the VQ-VAE model would lead to accessing the training data twice an epoch, thereby consuming a significant portion of the privacy budget ϵ. To resolve this issue, we propose the purple line, which represents the deployment of the VQ-VAE model with the DP. This approach accesses training data only once per epoch during the training stage.

4.4.2 Implementation for Training the VQ-VAE with the DP

Algorithm 9 describes how to train a VQ-VAE model under the DP in standard, and instead, Algorithm 10 shows the revised training flow for integrating the VQ-VAE with the DP. The core of the Algorithm 10 compared with Algorithm 9 is to combine the two gradient updates, which are separate for the AutoEncoder and PixelCNN models, into a single one. This reduces the time of accessing training data, thereby achieving the goal of lowering privacy loss. To reduce the privacy budget ϵ, we propose a modified training flow optimized explicitly for the VQ-VAE model. This method ensures that only one access of the original data is needed to train both the VQ-VAE and PixelCNN models during each training epoch, rather than training

Algorithm 9. Deploy vanilla training for the VQ-VAE model (VTF)

Input: training data x, VQ-VAE m_1, PixelCNN m_2, epoch T

Output: Reconstructed data y_1, Synthetic data y_2

procedure TRAINING FOR THE VQ-VAE WITH THE DP(x, m_1, m_2)
    $m_1, m_2 \leftarrow$ DP injection
    for epoch $t \leftarrow 1$ to T **do**
        $x_{enc} \leftarrow m_1.Encoder(x)$
        $x_{ind} \leftarrow m_1.Codebook(x_{enc})$
        $y_1 \leftarrow m_1.Decoder(x_{ind})$
        Backward propagation $\leftarrow m_1.loss(x, y_1)$
    end for
    for epoch $t \leftarrow 1$ to T **do**
        $x_{enc} \leftarrow m_1.Encoder(x)$
        $x_{ind} \leftarrow m_1.Codebook(x_{enc})$
        $y_{ind} \leftarrow m_2(x_{ind}))$
        Backward propagation $\leftarrow m_2.loss(x_{ind}, y_{ind})$
    end for
    $y_2 \leftarrow m_1.Decoder(m_1.Codebook(m_2(noise)))$
end procedure

them separately twice.

4.4.3 Description for Metrics

We evaluate models based on several metrics, including Inception Score (IS) [106], Fréchet Inception Distance (FID) [107], Peak Signal to Noise Ratio (PSNR) [108], Latency, and privacy budget Epsilon ϵ [109]. The results are presented as mean values with their respective standard deviations in parentheses. Using multiple metrics will allow for a more comprehensive, robust, and balanced assessment of our privacy-preserving generative model. It can also facilitate comparison with different models in the field and provide a richer understanding of its performance

Algorithm 10. Deploy revised training for the VQ-VAE model with the DP (RTF)

Input: training data x, VQ-VAE m_1, PixelCNN m_2, epoch T

Output: Reconstructed data y_1, Synthetic data y_2

procedure TRAINING FOR THE VQ-VAE WITH THE DP(x, m_1, m_2)
    $m_1, m_2 \leftarrow$ DP injection
    for epoch $t \leftarrow 1$ to T **do**
        $x_{enc} \leftarrow m_1.Encoder(x)$
        $x_{ind} \leftarrow m_1.Codebook(x_{enc})$
        $y_1 \leftarrow m_1.Decoder(x_{ind})$
        $y_{ind} \leftarrow m_2(x_{ind}))$
        Backward propagation $\leftarrow m_1.loss(x, y_1), m_2.loss(x_{ind,y_{ind}})$
    end for
    $y_2 \leftarrow m_1.Decoder(m_1.Codebook(m_2(noise)))$
end procedure

under different conditions or aspects.

The IS and FID are both widely used objective metrics for evaluating the quality of generated images, particularly those produced by generative models. The IS measures the diversity and quality of generated images by assessing the output distribution of a pre-trained Inception model [110]. A higher IS value indicates better performance, suggesting that the generative model has produced images with diverse and recognizable features. On the other hand, the FID compares the distribution of generated images with the distribution of a set of real images, often referred to as the "ground truth." This metric calculates the Fréchet distance [111] between activation values of the generated images and the real images in the latent or feature space of a pre-trained Inception model provided by Google. A lower FID value indicates better performance, suggesting that the generative model's output is closer to the real image distribution.

The PSNR is another commonly used metric for evaluating the quality of generated images, especially in the context of image reconstruction. The PSNR is derived from Mean Squared Error (MSE) [112] between the original and the generated or

reconstructed images. It measures the ratio between the maximum possible power of the signal, i.e., the image's pixel intensity, and the power of the corrupting noise, i.e., the error introduced during image generation or reconstruction. The number is expressed in decibels (dB), and a higher PSNR value typically indicates better image quality, as it represents a more negligible difference between the generated or reconstructed image and the original image. In other words, a higher PSNR suggests that the generative model has successfully captured the essential features of the original image with less distortion or noise, which serves as a useful quantitative measure to compare the performance of different generative models in terms of their ability to maintain image fidelity considering whether to deploy the DP.

Latency of training and privacy budget ϵ are also important metrics to consider when evaluating the performance of privacy-preserving generative models, as they provide insights into the efficiency and privacy guarantees of the models. The latency measures the time that it takes for the VQ-VAE model to produce an output, such as generating a synthetic image or reconstructing an input image. Latency is typically expressed in seconds and can be crucial in assessing the real-world applicability of a generative model, especially in scenarios where real-time or near-real-time processing is required. A lower latency value indicates faster performance, making the model more suitable for time-sensitive applications.

In the context of the DP, the Epsilon ϵ quantifies the level of privacy protection provided by a privacy-preserving algorithm. A smaller Epsilon value signifies stronger privacy guarantees, implying that the algorithm's output is less sensitive to changes in individual data points. However, achieving a lower Epsilon often comes at the cost of reduced utility or performance, as more noise is introduced to preserve privacy. The amount of noise depends on the sensitivity [113] of the function and is drawn from a distribution. The sensitivity of a function indicates how much the output might vary in response to changes in the input. Specifically, it represents the most tremendous possible change in the output when a single individual is either added to or removed from any potential input dataset. Sensitivity is a crucial concept as it guides the amount of noise required to be introduced to the function's output to safeguard the privacy of individuals in any potential input dataset. A higher

sensitivity necessitates the addition of more noise.

4.5 Evaluation

We carry out a comprehensive series of experiments to evaluate the effectiveness of our proposed modified training flow for the VQ-VAE model with the DP. These experiments are meticulously designed to measure both the quality of the image generation produced by the model and the level of privacy-preserving afforded by the training method. We evaluate the model using diverse datasets and various performance metrics and compare it with the RTF against the standard training method VTF to ensure a rigorous and exhaustive assessment.

In the VQ-VAE, the prior distribution over the discrete latent is a categorical distribution and can be made autoregressive by depending on other vectors in the feature map. While training the VQ-VAE, the prior is kept constant and uniform. After training, we fit an autoregressive distribution over indices obtained from the codebook to generate data via ancestral sampling. We use a PixelCNN model over the discrete latent for the current work. The PixelCNN model is a type of Convolutional Neural Network (CNN) model capable of generating images pixel by pixel, conditioned on the previous pixels. It's called "autoregressive" because each pixel is a function of the previous ones, which is to learn a distribution over the latent that can be sampled from to generate new data.

4.5.1 Experimental Setup

We conduct our experiments using Python 3.9.11 [83]. The VQ-VAE and PixelCNN models are developed using TensorFlow 2.10.0 [114].

Datasets. We utilize popular image datasets, including Modified National Institute of Standards and Technology (MNIST) [115], the Fashion-MNIST [116], and Canadian Institute for Advanced Research 10 classes (CIFAR-10) [117], to train the VQ-VAE and PixelCNN models, both with and without the DP. The MNIST and Fashion-MNIST are original $60,000$ of 28×28 grayscale images in both the training and test datasets, and the CIFAR-10 is $60,000$ of 32×32 color image dataset for

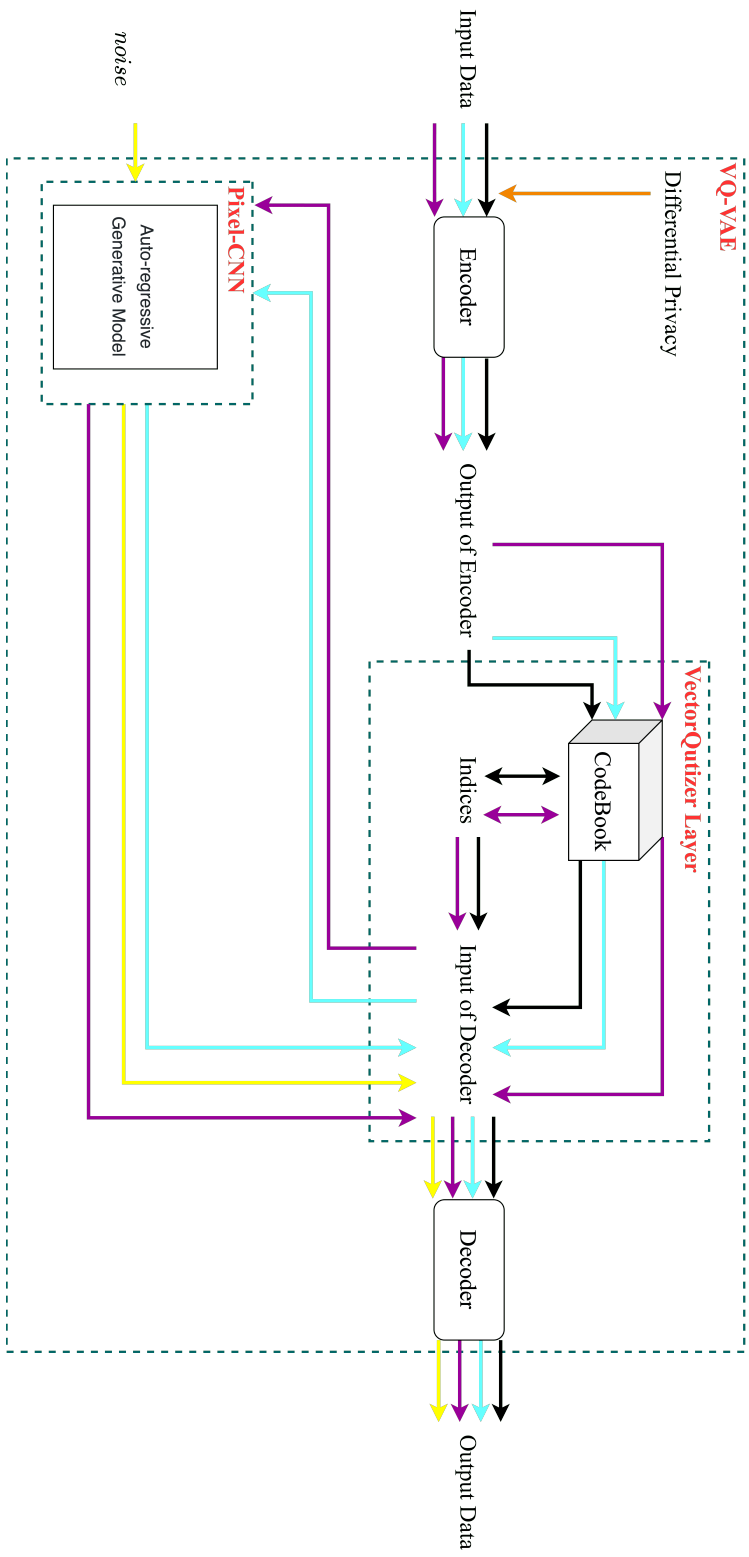


Figure 4.2. The data flow for the VQ-VAE model. The black line is the flow for the VQ-VAE model, the blue line is the flow for the PixelCNN model, the yellow line is the flow for sampling, and the purple line is the flow for the proposed training flow of the VQ-VAE.

10 classes with 50,000 training and 10,000 test images per class. These datasets primarily focus on supervised learning, but our objective centers around generative models, particularly unsupervised learning. Consequently, we use the STL [118] dataset, specifically designed for image recognition tasks and intended to facilitate the development of the unsupervised DL approaches. This setup allows us to explore the potential benefits of the unsupervised and self-supervised DL techniques that leverage a large amount of unlabeled data to improve model performance in scenarios with limited labeled data.

Hardware and Software. Throughout our experiments, we utilize two different devices for vanilla test and final validation purposes. A MacBook Pro is employed for the test phase, featuring an M1 chip and RAM 16GB. For the validation phase, we use a laptop running Windows 11, which is equipped with 12th Gen Intel(R) Core(TM) i7-12700H 2.30GHz, RAM 32GB, NVIDIA GeForce RTX 3080. This setup allows us to verify the consistency and compatibility of our results across different operating systems and hardware configurations. All experimental results are logged and recorded using the laptop with the operating system Windows 11.

Our experiments use Python 3.9 [83] to code and implement our methods. We build the VQ-VAE and PixelCNN models using the TensorFlow library 2.10.0 [114], a popular open-source framework for DL applications. This choice provides us with the necessary tools and flexibility to develop and test our models efficiently.

For deploying the DP, we utilize the TensorFlow Privacy library 0.8.7 [29], an extension of TensorFlow that provides privacy-preserving mechanisms for the DL. The TensorFlow Privacy library enables us to integrate differential privacy techniques into our models, ensuring that privacy constraints are met throughout the training process. The main processes to guarantee the DP were done by the TensorFlow Privacy library. The privacy budget ϵ is computed by the total number of points in the training or test data, batch size, the number of epochs of training or test, and delta δ, which is set to be less than the inverse of the size of the training or test dataset.

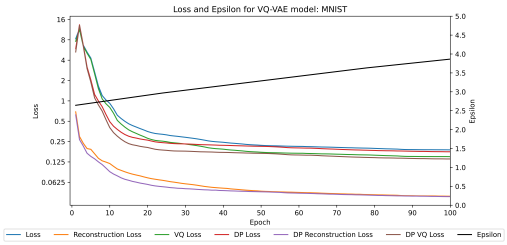
Metrics. We evaluate our models using several metrics, including the IS, FID, and PSNR. Additionally, we assess training latency and consider the privacy budget ϵ

consumed during the training process. The numerical data of the training trends are represented in Figure 4.3, which depicts the trend in training losses for four distinct datasets using the VTF and RTF. Moreover, it shows the consumption of the privacy budget ϵ throughout the process in the DP scenario. We implement the TensorFlow Privacy library to calculate the privacy budget and add noise applied to the gradient during forward propagation. Whether adding or deleting an image, the difference is always one, and the sensitivity is one.

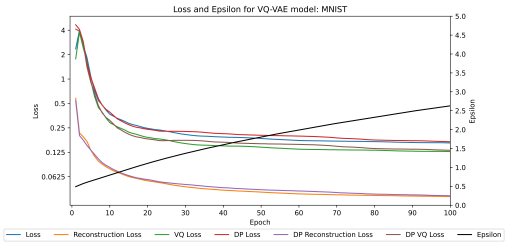
4.5.2 Optical Comparison in Latent Space for Processed Data

To observe and compare the feasibility of the proposed method more intuitively, we will visualize the latent space variables for comparisons at the human eye level. The depicted latent figures in the middle column of Figure 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, and 4.12 offer a more in-depth look at the data gathered during our experiment, further supporting and enhancing the findings. First, we show the ground truth images in Figure 4.4 for deploying the VQ-VAE model with and without the DP in the VTF and RTF for four datasets. Each category of the dataset shows 30 images representing samples.

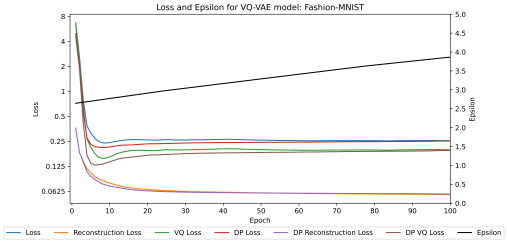
In detail, the image results are divided into eight figures; each of these figures is distinct based on the combination of the three attributes: the DP application with affirmative or negative, method type with the VTF or RTF, and dataset coloration with grayscale or color. Figure 4.5 includes images that have not applied the DP, utilized the VTF method, and belong to the grayscale images. Figure 4.6 comprises images that have not applied the DP, implemented the VTF, and belong to the color images. In Figure 4.7, images are processed by the DP, generated by the VTF method, and are part of grayscale images. In Figure 4.8, images are operated by the DP, executed by the VTF method, and are part of color images. On the contrary, images displayed in Figure 4.9 consist of results without the DP, realized by the RTF method, and are from the grayscale images. Moreover, images illustrated in Figure 4.10 consist of results without the DP, accomplished by the RTF method, and are from the color images. Furthermore, Figure 4.11 contains images carried out by the DP, put into action of the RTF method, and are the grayscale dataset. Finally,



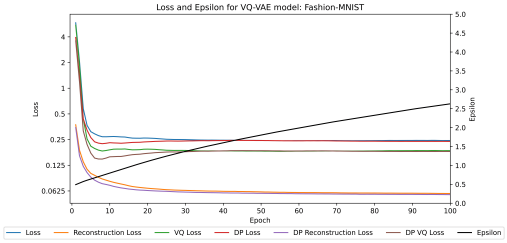
(a). Loss and epsilon of the MNIST in VTF



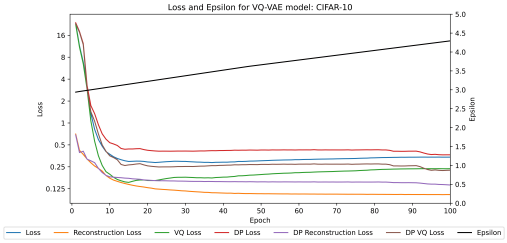
(b). Loss and epsilon of the MNIST in RTF



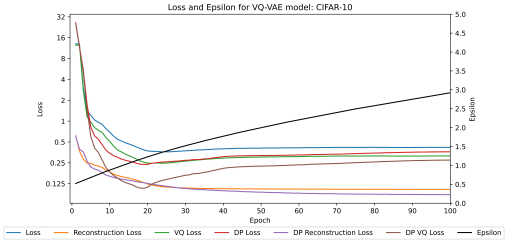
(c). Loss and epsilon of the Fashion-MNIST in VTF



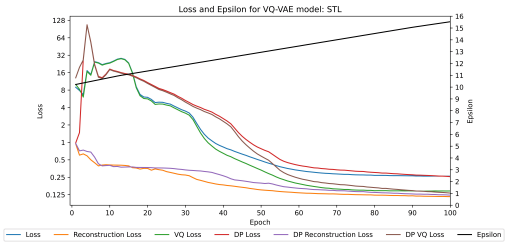
(d). Loss and epsilon of the Fashion-MNIST in RTF



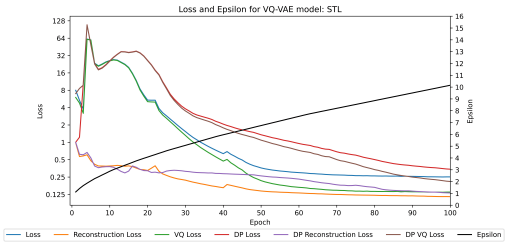
(e). Loss and epsilon of the CIFAR-10 in VTF



(f). Loss and epsilon of the CIFAR-10 in RTF



(g). Loss and epsilon of the STL in VTF



(h). Loss and epsilon of the STL in RTF

Figure 4.3. Visualizing loss and epsilon for the VQ-VAE with and without the DP on datasets. Unraveling the learning dynamics of discrete representation. Under the RTF, it can be clearly observed that each dataset accumulates less privacy budget as the training epochs increase in the black line compared with the VTF.

Figure 4.12 encloses images enforced by the DP, put into action of the RTF method, and are the color images.

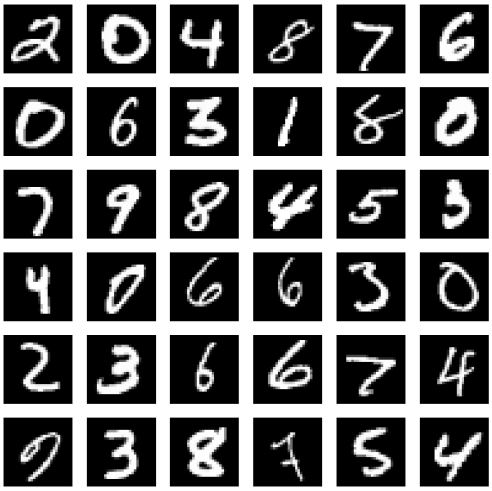
During the visualization images, from Figure 4.5 to Figure 4.12, there are three parts presenting optical results. The left part is generated by the VQ-VAE model, which is the generation result and has 30 generation images. Specifically, to show the comparison among ground truth, generation, and latent space, 10 combinations of images of the middle part are illustrated in order of ground truth, generation, and latent space. Meanwhile, the VQ-VAE model includes the generation and the sampling; in addition to presenting the generation results, it is also necessary to show the sampling results. As for the sampling, 10 groups of latent space and images sampled from noise for the sampling are displayed in the right part.

4.5.3 Implement the VQ-VAE Model with and without the DP in the Vanilla Training Flow

In this subsection, we discuss the implementation of the VQ-VAE model with and without the DP using the VTF, which is illustrated in from Figure 4.5 to Figure 4.8.

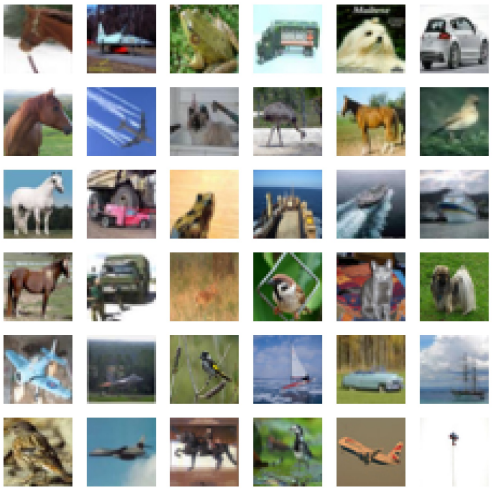
Implementing the VQ-VAE model without the DP in the VTF. To implement the VQ-VAE model without the DP, we follow the VTF. Figure 4.5 and Figure 4.6 display the results of grayscale and color datasets, respectively. This experiment follows the standard training method for the VQ-VAE model without any modification or the DP; in other words, it trains the encoder and the decoder of the VQ-VAE model first, then trains the AR model tuning the vector quantized layer of the model and the decoder. The entire flow is depicted in the order of the black line and the blue line of Figure 4.2.

Implementing the VQ-VAE model with the DP in the VTF. To implement the VQ-VAE model with the DP, we leverage the TensorFlow Privacy library, an extension of the TensorFlow that provides a suite of DP preserving mechanisms. The images results are shown in Figure 4.7 and Figure 4.8. This experiment verifies how to implement the DP into the training of the VQ-VAE model. Furthermore, the exact flow is depicted in Figure 4.2, including the orange line for the DP, the black line for

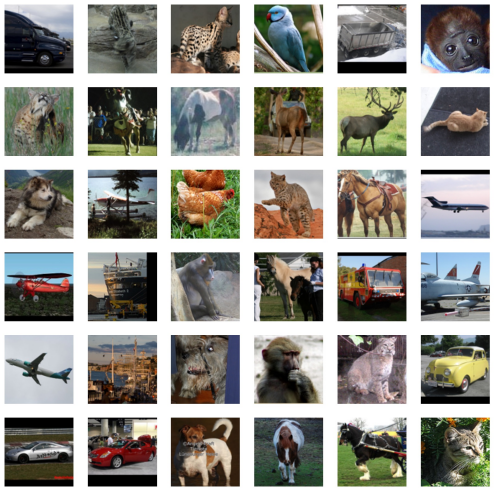


(a). Ground truth of the MNIST

(b). Ground truth of the Fashion-MNIST



(c). Ground truth of the CIFAR-10



(d). Ground truth of the STL

Figure 4.4. Ground truth images for grayscale datasets the MNIST and Fashion-MNIST, and color datasets the CIFAR-10 and STL.

the VQ-VAE model, and the blue line for the AR model.

4.5.4 Implement the VQ-VAE Model with and without the DP in the Revised Training Flow

In this subsection, we will discuss the revised method for deploying the VQ-VAE model with and without the DP, which reduces the privacy budget ϵ, enhancing privacy-preserving. Moreover, it conducted a comparative experiment between the VAE and the VTF, demonstrating improved performance when implementing DP in the VQ-VAE model as opposed to the VAE model. The result can be seen in Figure 4.13. Furthermore, the optical results are illustrated from Figure 4.9 to Figure 4.12.

Implementing the VQ-VAE model without the DP in the RTF. Before deploying the revised training flow with the DP in the VQ-VAE model, it is necessary to test its feasibility in a non-DP environment. This step is crucial to evaluate the general applicability of the proposed method, which validates whether the proposed method RTF shows better performance, especially privacy budget ϵ, than the VTF in the DP environment. The results are clearly presented in Figure 4.9 and Figure 4.10.

Implementing the VQ-VAE model with the DP in the RTF. The primary goal of incorporating the DP into the VQ-VAE model is to preserve the privacy of the input data while still generating high-quality images. The DP mechanism ensures that the model learns the global patterns and structures from the dataset without memorizing specific details of individual samples. Furthermore, as previously mentioned, privacy budget ϵ is critical for privacy-preserving; the lower value of privacy budget ϵ provides a higher level of privacy-preserving. Thus, reducing the privacy budget while holing, to some extent, performance is the main objective of this contribution, which is ensured by the proposed RTF method. Similarly, the image results have been depicted in Figure 4.11 and Figure 4.12. Moreover, the elaborated flow is drawn in Figure 4.2 with the purple line for the proposed RTF method, the orange line for the DP, and the yellow line for the sampling.

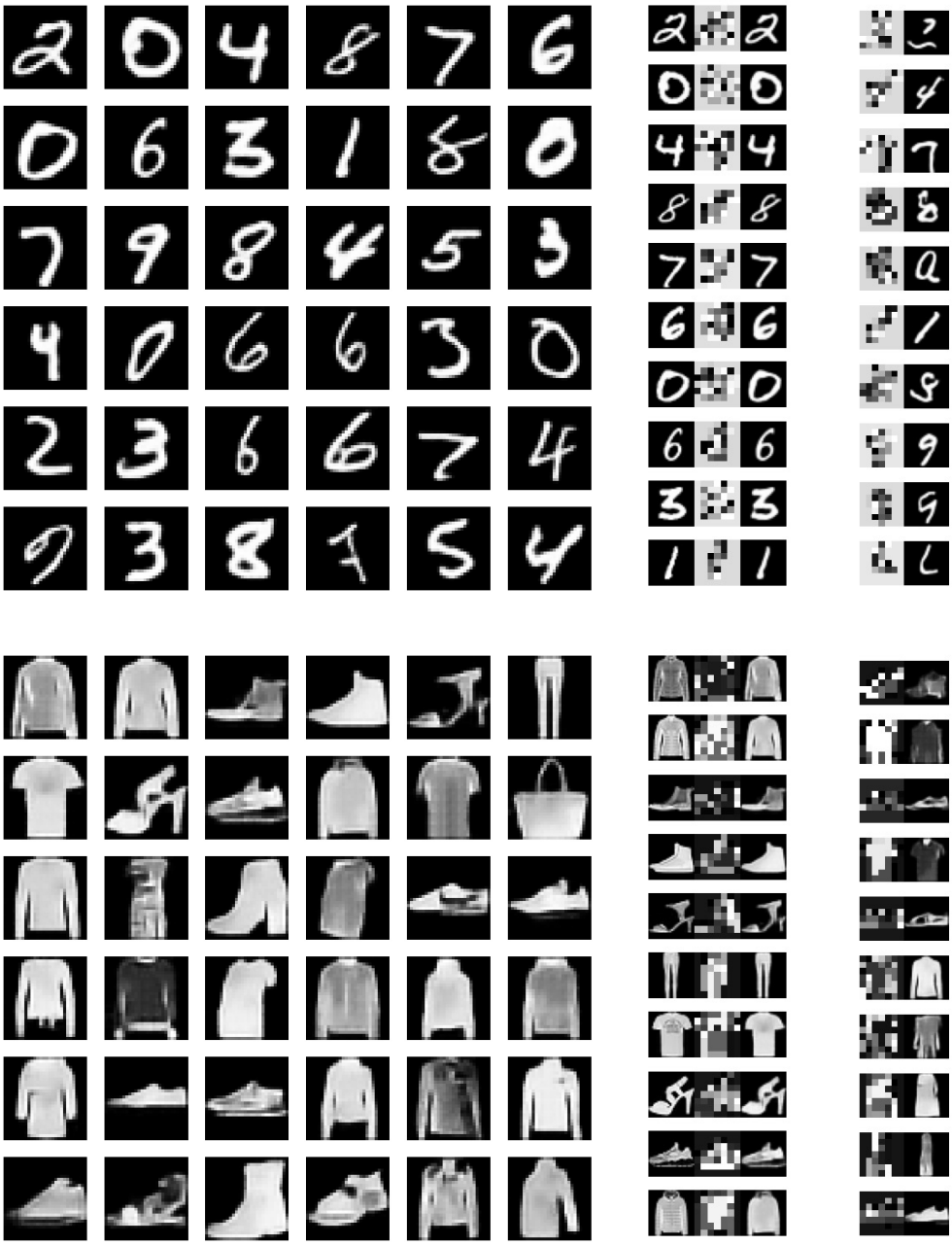


Figure 4.5. Visualizing latent space for deploying the VQ-VAE model without the DP in the VTF for grayscale datasets the MNIST of the upper part, Fashion-MNIST of the lower part in reconstruction and sampling. The left column is the input, and the middle column is a combination of input, latent space, and reconstructed images. The right column is the combination of latent space and sampled images.

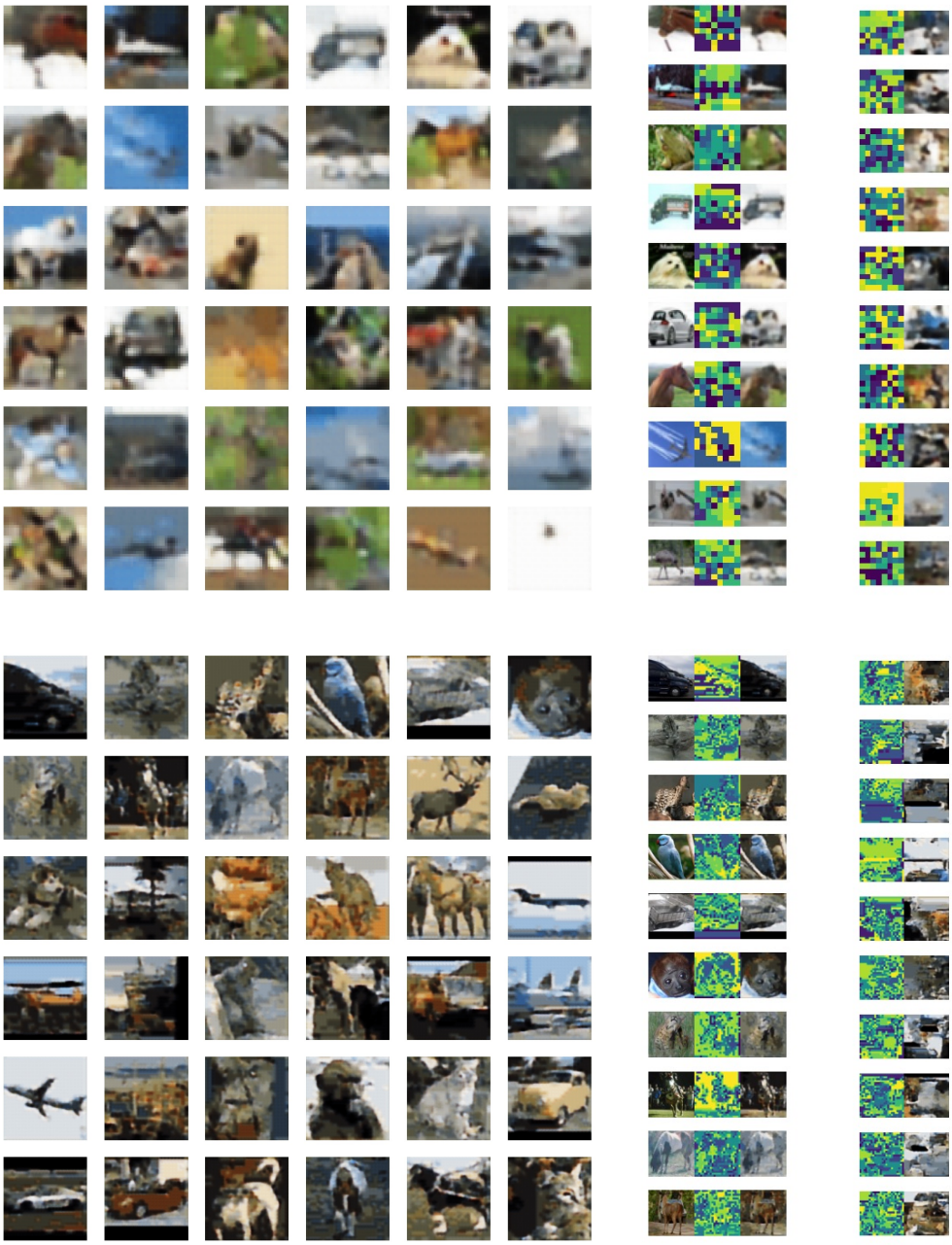


Figure 4.6. Visualizing latent space for deploying the VQ-VAE model without the DP in the VTF for color datasets CIFAR-10 of the upper part, STL of the lower part in reconstruction and sampling. The left column is the input, and the middle column is a combination of input, latent space, and reconstructed images. The right column is the combination of latent space and sampled images.

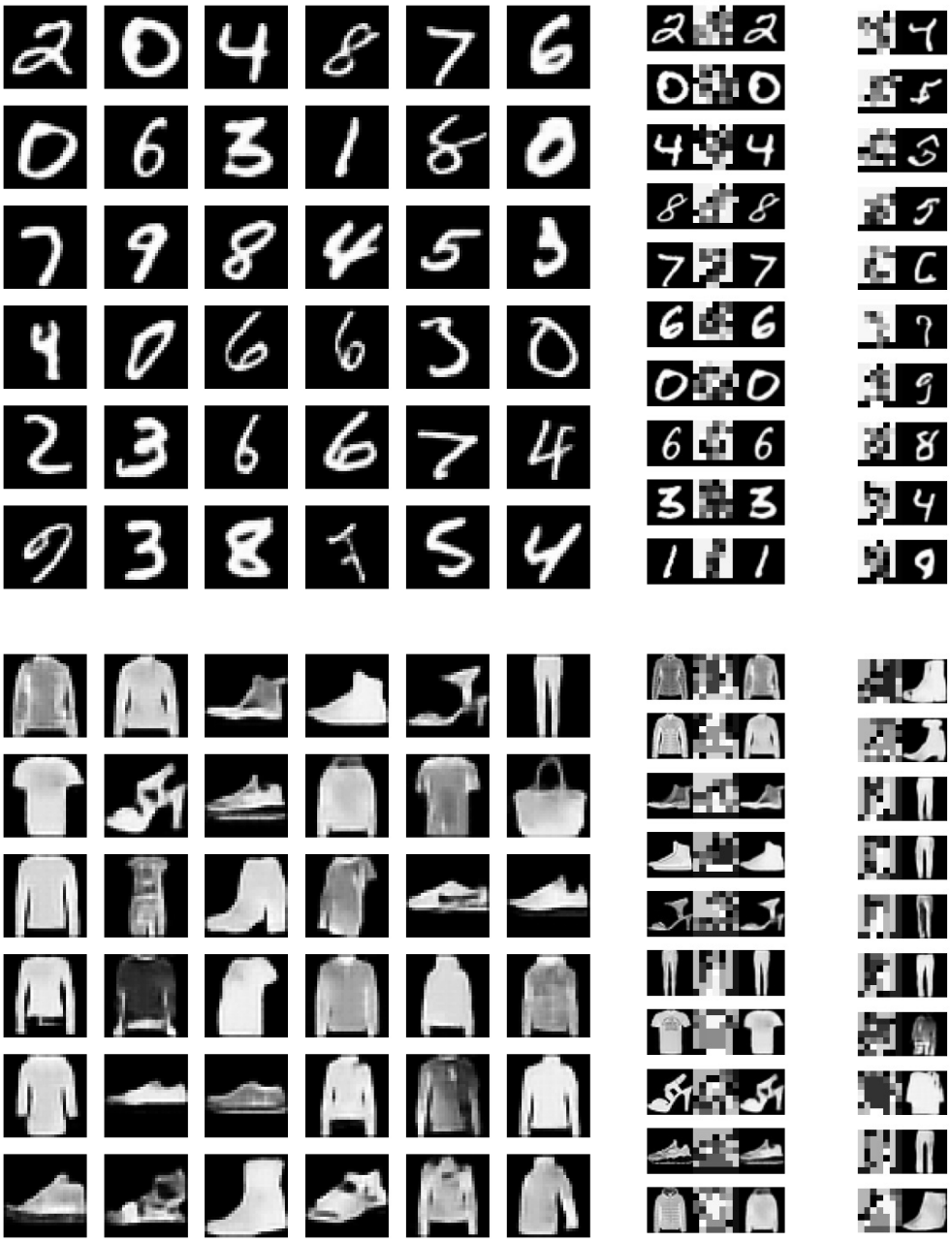


Figure 4.7. Rendering latent space for the implementation of the VQ-VAE model with the DP in the VTF for grayscale datasets like the MNIST of the upper part and Fashion-MNIST of the lower part in both reconstruction and sampling. The left column is the input, and the middle column is a combination of input, latent space, and reconstructed images. The right column is the combination of latent space and sampled images.

Figure 4.8. Rendering latent space for the implementation of the VQ-VAE model with the DP in the VTF for color datasets like the CIFAR-10 of the upper part and STL of the lower part in both reconstruction and sampling. The left column is the input, and the middle column is a combination of input, latent space, and reconstructed images. The right column is the combination of latent space and sampled images.

Figure 4.9. Executing latent space visualization for the activation of the VQ-VAE model devoid of the DP within the RTF pertaining to grayscale datasets such as the MNIST of the upper part and Fashion-MNIST of the lower part for reconstruction and sampling. The left column is the input, and the middle column is a combination of input, latent space, and reconstructed images. The right column is the combination of latent space and sampled images.

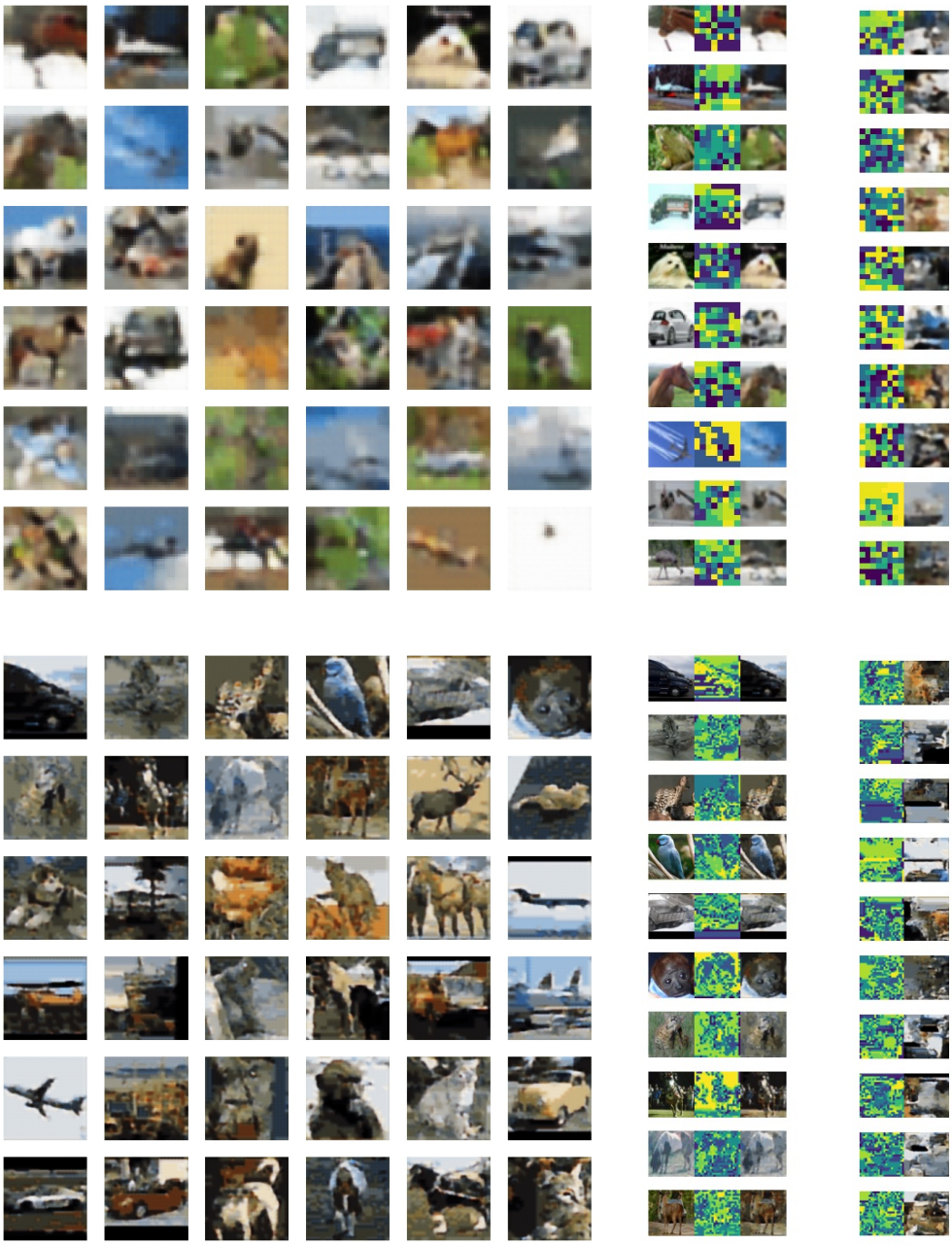


Figure 4.10. Executing latent space visualization for the activation of the VQ-VAE model devoid of the DP within the RTF pertaining to color datasets such as the CIFAR-10 of the upper part and STL of the lower part for reconstruction and sampling. The left column is the input, and the middle columnais the combination of input, latent space, and reconstructed images. The right column is the combination of latent space and sampled images.

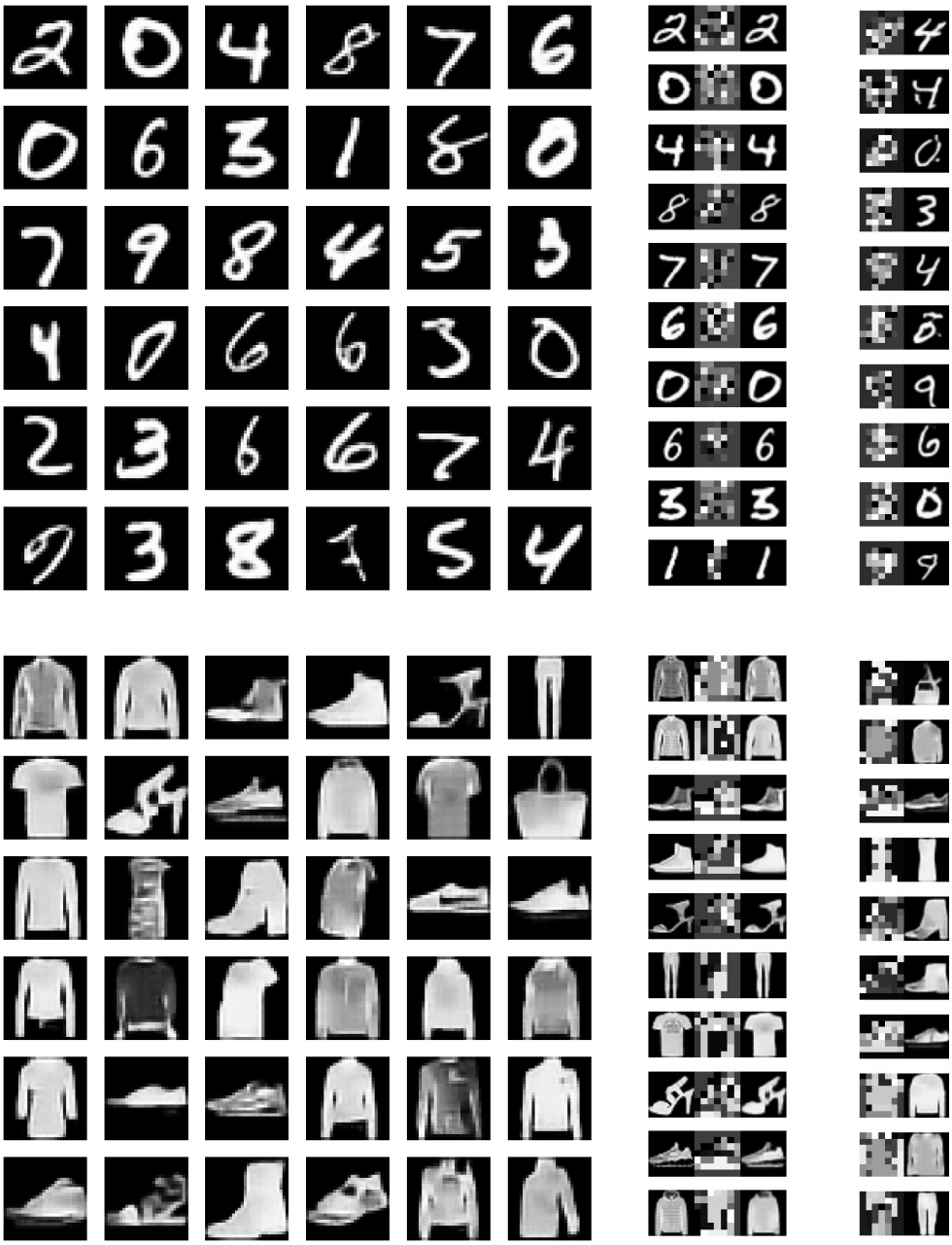


Figure 4.11. Illustrating latent space for executing the VQ-VAE model with the DP in the RTF for grayscale datasets, including the MNIST of the upper part and Fashion-MNIST of the lower part, in reconstruction and sampling. The left column is the input, and the middle column is a combination of input, latent space, and reconstructed images. The right column is the combination of latent space and sampled images.

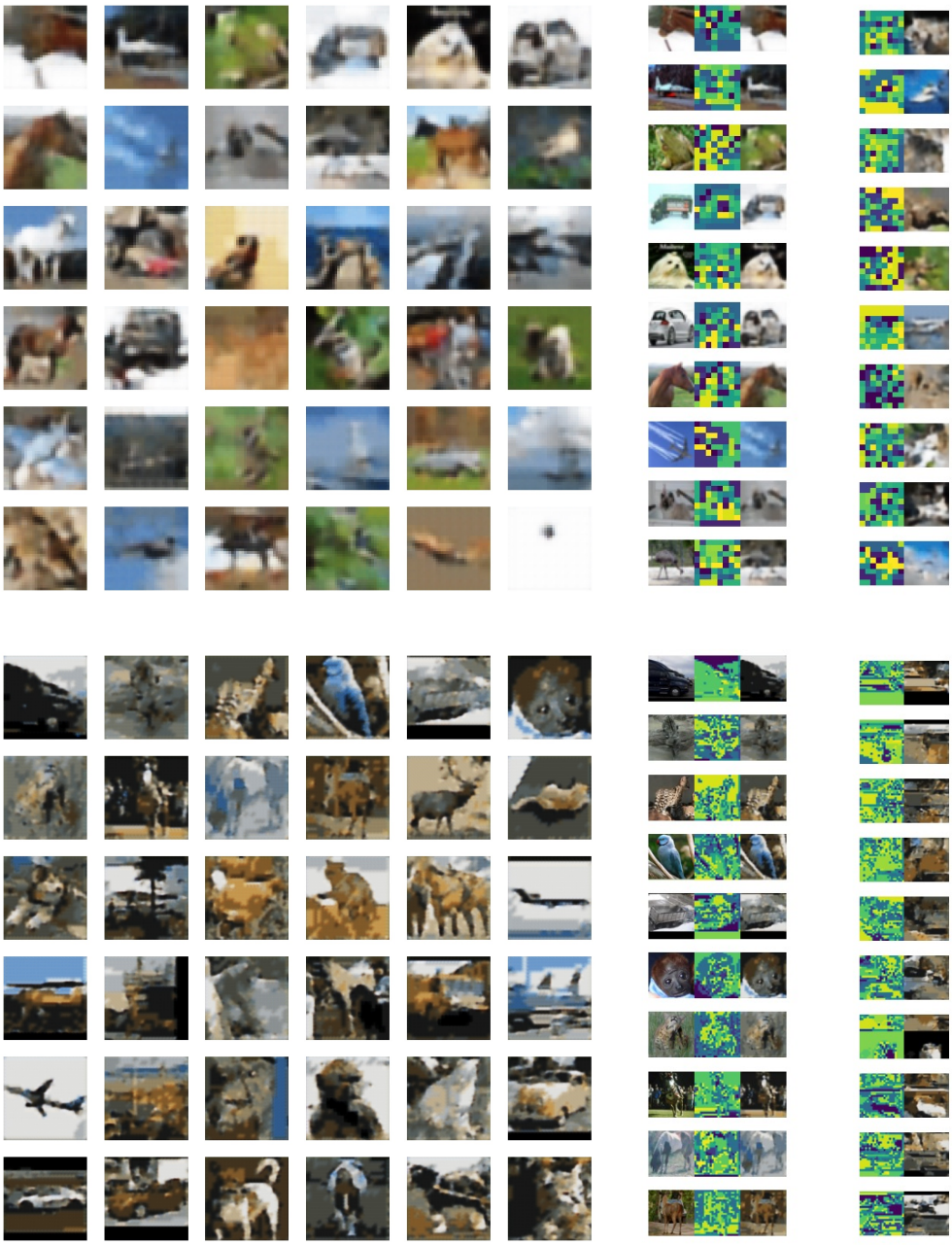


Figure 4.12. Illustrating latent space for executing the VQ-VAE model without the DP in the RTF for color datasets, including the CIFAR-10 of the upper part and STL of the lower part, in reconstruction and sampling. The left column is the input, and the middle column is a combination of input, latent space, and reconstructed images. The right column is the combination of latent space and sampled images.

4.5.5 Results Analysis

By observing the first and second columns of each evaluation metric in Figure 4.13, we can see that the performance of the VQ-VAE models using the DP is superior to that of the VAE models using the DP. This demonstrates that by applying the DP, we can still achieve better performances on the VQ-VAE models than on the VAE models.

We trained the VQ-VAE model without the DP in the VTF, which helps it to understand and learn the underlying data structure and the complex relationships between various data features. Once the VQ-VAE model had been well-trained, we processed to train a prior model, the PixelCNN model. We trained the PixelCNN on the discrete latent codes obtained from the well-trained VQ-VAE model. This process is crucial because it allows the PixelCNN to learn the distribution of these latent codes, enabling it to generate new ones that are likely to occur based on its training. Consequently, this makes it possible to generate synthetic images from the combined the VQ-VAE and PixelCNN models.

The training process of the RTF is refined to accommodate the requirements of the DP, which includes the calculation of per-sample gradients, the implementation of gradient clipping, and the injection of noise. To achieve privacy-preserving, we utilized the DP-SGD optimizer, which calculates the average of clipped gradients and adds Gaussian noise to the gradients before applying parameter updating. It ensures that the model's updates do not reveal any sensitive information about the input data. In the DP-SGD, noise is added to the gradient computation during the training process to ensure differential privacy. This is typically done in two steps:

I First, the gradients, which are derived from the training data, for a batch of data are computed.

II Then, before applying these gradients to update the model parameters, an appropriate amount of random noise is added to these gradients.

This noise ensures that the influence of any individual data point on the model's parameters is limited, providing privacy guarantees. With the revised training pro-

cess utilized with the DP, it is possible to train the VQ-VAE model in a better privacy-preserving manner, which is shown by the purple line in Figure 4.2. The training method of the VQ-VAE with the DP in the RTF remains essentially unchanged, with the primary difference being the integration of the DP-SGD into the backpropagation step, being able to train a model that protects the privacy of the input data while still learning effectively from it.

The image data shows the results of the training data under reconstruction and sampling and presents the corresponding relationship of the latent space and distribution map learned through the autoencoder. In addition, we display the comparison diagram of the relationship between the latent space and the output under random sampling. Through the image results, it can be clearly seen that the application of the DP will not visually change the data too much. Still, from the perspective of the data itself, sensitive privacy data has been removed, ensuring the privacy and security of the output data. This also proves the effectiveness of the method we proposed.

In our experimental results, as shown in Table 4.1 with separated analysis of epsilon for Table 4.2 and latency for Table 4.3, Figure 4.3, and Figure 4.13. To highlight the effectiveness of the data, in Table 4.1, we conducted comparisons for the same dataset under different schemes, different methods, and different evaluation criteria. For the same dataset, data with a green background indicates superior performance in different schemes; under different methods and the same evaluation criterion, data in red indicates more excellent performance. Furthermore, we evaluate the performance across three dimensions: horizontal, vertical, and privacy-preserving.

In horizontal comparison, by analyzing the results of each dataset under different methods, the VTF and RTF, we can conclude that in the regular scheme, changing the training method does not have a particularly noticeable effect on the final performance, and it may even cause a decline. In the DP scheme, we can see a significant improvement in the results, which indicates that our proposed method can effectively enhance the model's performance under the DP scheme. However, it should be noted that the final training latency is significantly lower for the VTF compared to the VTF, as the VTF only calls the original dataset once during each

training epoch.

In vertical comparison, for the VTF, by applying the DP scheme, we can observe a noticeable decline in the results, as the DP introduces noise that affects the final performance. On the other hand, for the RTF, after applying the DP scheme, the performance has significantly improved, which demonstrates that our proposed method has excellent adaptability for the DP scheme.

In privacy-preserving comparison, we calculate the privacy budget ϵ for the four datasets under the two methods. In the VTF, the privacy budgets consumed for the MNIST, Fashion-MNIST, CIFAR-10, and STL are 3.86, 3.86, 4.94, and 15.53, respectively. In the RTF, the privacy budgets consumed for the MNIST, Fashion-MNIST, CIFAR-10, and STL are 2.63, 2.63, 2.92, and 10.16, respectively. By calculating the differences, it can be concluded that deploying the DP-enabled VQ-VAE model using our proposed method RTF can reduce the privacy budgets by 31.87%, 31.87%, 40.89%, and 34.58% on the corresponding datasets. In other words, the proposed method RTF can reduce the privacy budget ϵ by approximately 34.80% while maintaining the performance, resulting in a more robust privacy-preserving level.

In Figure 4.13, we evaluate three models and methods, the VAE, VTF, and RTV, on four datasets using both plaintext data and the DP-processed data. The test metrics include the IS, which gets the higher the score, the better; the PSNR, which obtains the higher the score, the better; and the FID, which performs the lower the score, the better. Based on the experimental results, although the RTV model proposed in this chapter does not achieve better results on plaintext data, it shows outstanding performance on all four metrics across the four datasets when using the DP-processed data. Our results suggest that the RTF strikes an effective balance between privacy-preserving and image-generation performance. By reducing the privacy budget ϵ without compromising the quality of reconstructed and generated images, the RTF exhibits significant potential for various applications that demand high-quality image generation coupled with robust privacy guarantees.

Table 4.1. Numerical comparison of the previous method VTF and the proposed method RTF on four datasets. A green background indicates better performance in vertical comparison, while red text indicates better performance in horizontal comparison.

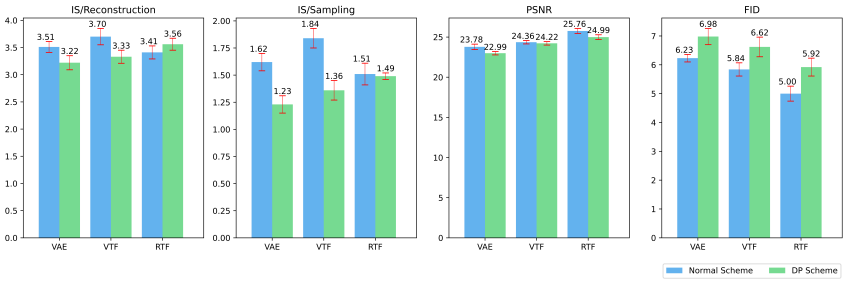
Method														
Dataset	VTF						RTF							
	IS/Reconstruction	FID	PSNR	IS/Sampling	Latency (second)	Epsilon	Scheme	IS/Reconstruction	FID	PSNR	IS/Sampling	Latency (second)	Epsilon
MNIST	3.70 (±0.15)	5.84 (±0.23)	24.36 (±0.21)	1.84 (±0.23)	22.42 (±0.22)	-	Normal	3.41 (±0.12)	5.00 (±0.26)	25.76 (±0.31)	1.51 (±0.10)	15.19 (±0.45)	-
	3.33 (±0.12)	6.62 (±0.34)	24.22 (±0.22)	1.36 (±0.09)	20.09 (±0.37)	3.86	DP	3.56 (±0.11)	5.92 (±0.31)	24.99 (±0.30)	1.49 (±0.03)	14.91 (±0.38)	2.63
Fashion-MNIST	2.16 (±0.10)	16.20 (±0.30)	21.59 (±0.19)	2.24 (±0.17)	21.01 (±0.36)	-	Normal	1.92 (±0.13)	15.40 (±0.23)	22.21 (±0.35)	1.17 (±0.21)	15.05 (±0.44)	-
	2.05 (±0.13)	16.99 (±0.29)	21.52 (±0.28)	1.17 (±0.21)	22.01 (±0.28)	3.86	DP	2.31 (±0.13)	15.83 (±0.34)	22.05 (±0.32)	2.36 (±0.13)	15.12 (±0.37)	2.63
CIFAR10	3.18 (±0.20)	81.69 (±1.00)	22.46 (±0.39)	4.27 (±0.13)	24.53 (±1.28)	-	Normal	2.82 (±0.16)	77.92 (±1.14)	23.11 (±0.23)	4.08 (±0.06)	13.79 (±1.72)	-
	2.71 (±0.18)	89.03 (±2.11)	21.12 (±0.41)	4.35 (±0.11)	24.36 (±1.75)	4.94	DP	3.09 (±0.19)	73.34 (±1.34)	22.94 (±0.27)	4.73 (±0.22)	13.75 (±1.34)	2.92
STL	2.64 (±0.19)	66.41 (±3.01)	21.52 (±0.33)	2.64 (±0.68)	22.78 (±1.21)	-	Normal	1.51 (±0.21)	61.17 (±1.98)	23.04 (±0.29)	1.33 (±0.31)	8.56 (±1.11)	-
	1.51 (±0.22)	65.48 (±2.78)	21.30 (±0.27)	2.35 (±0.20)	22.81 (±1.18)	15.53	DP	1.85 (±0.18)	57.95 (±2.31)	23.22 (±0.34)	2.69 (±0.22)	8.83 (±1.23)	10.16

Table 4.2. Quantitative analysis of epsilon. The RTF shows a lower privacy budget compared with the VTF in different datasets.

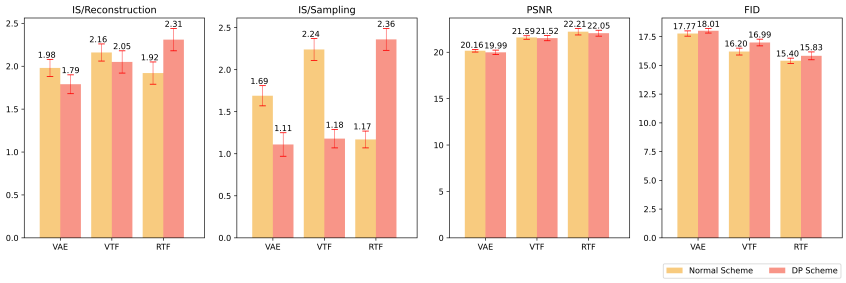
		Method	
Dataset	Scheme	VTF	RTF
		Epsilon	
MNIST	Normal	-	-
	DP	3.86	2.63
Fashion-MNIST	Normal	-	-
	DP	3.86	2.63
CIFAR10	Normal	-	-
	DP	4.94	2.92
STL	Normal	-	-
	DP	15.53	10.16

Table 4.3. Summary of experimentally derived latency values with a certain degree of variability or precision around each value.

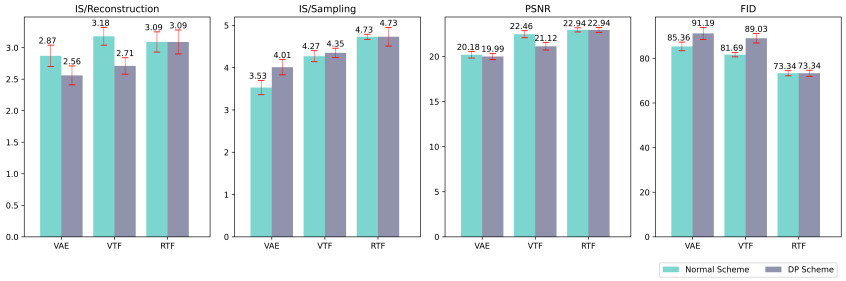
		Method	
Dataset	Scheme	VTF	RTF
		Latency (second)	
MNIST	Normal	22.42 (±0.22)	15.19 (±0.45)
	DP	20.09 (±0.37)	14.91 (±0.38)
Fashion-MNIST	Normal	21.01 (±0.36)	15.05 (±0.44)
	DP	22.01 (±0.28)	15.12 (±0.37)
CIFAR10	Normal	24.53 (±1.28)	13.79 (±1.72)
	DP	24.36 (±1.75)	13.75 (±1.34)
STL	Normal	22.78 (±1.21)	8.56 (±1.11)
	DP	22.81 (±1.18)	8.83 (±1.23)



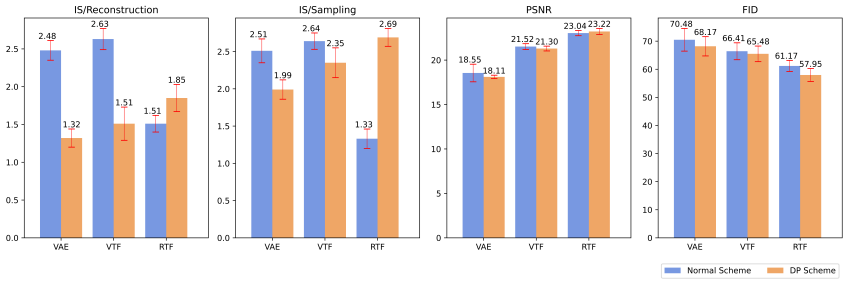
(a). Numerical results of implementing the MNIST



(b). Numerical results of applying the Fashion-MNIST



(c). Numerical results of deploying the CIFAR-10



(d). Numerical results of utilizing the STL

Figure 4.13. Numerical metrics results for four datasets: the MNIST, Fashion-MNIST, CIFAR-10, STL. The horizontal axis represents the model category, while the vertical axis represents the score.

4.6 Discussion

In the work of this chapter, we aim to apply the DP to a generative model of the server that executes the model, which is trusted, specifically the VQ-VAE, to generate synthetic images in a privacy-preserving manner. Indeed, the distributed approach can connect data islands, but to truly implement it requires a large amount of communication cost among people. Therefore, currently, the best way to obtain high-performance models is to collect and centrally process data obtained in various ways. Therefore, in our research, we implement the Gaussian-DP technology based on the Central-DP as a benchmark. This approach contrasts with directly applying the DP to input data, as it ensures privacy-preserving not just in the raw input data but in the synthetic data generated by the model. The rationale for focusing on the model rather than the data lies in the application of generative models. The primary purpose is to generate synthetic data that accurately mirrors the original data distribution without compromising the privacy of individuals whose data was used in the training process, consuming less privacy budget. We introduce a novel DP-enabled VQ-VAE method RTF that strikes a balance between preserving privacy and maintaining image-generation performance. Based on our experiment results, the method RTF we proposed has reduced the privacy budget by an average of 34.80% across four different datasets. This is a significant improvement from the VTF of the VQ-VAE.

The effectiveness of the proposed RTF in preserving privacy while generating high-quality images lends credence to the integration of privacy-preserving mechanisms, such as the DP, in the realm of generative models. Our work harmonizes with the body of existing literature advocating the need for such integration, challenging the conventional approaches and setting a new benchmark in privacy-preserving image generation tasks in a centralized scenario. The method's versatility across different datasets demonstrates its potential for broader applications in domains such as medical imaging, biometric systems, and personalized user experiences, where privacy preservation is critical.

Despite the promising results, some limitations still exist in the RTF. For instance, its scalability to larger and more complex datasets, the effect of the different generative models, and the impact of the decentralized implementation scenario on the privacy-utility trade-offs remain unexplored. Significantly, the DP scheme used in our work of this chapter is the central DP, and the local DP is a tendency for distributed computing. Additionally, integrating other privacy-preserving mechanisms, such as Secure Multi-party Computation (SMPC) or Homomorphic Encryption (HE), could strengthen the privacy guarantees of the method, opening up new possibilities for its application. The single privacy-preserving mechanism has application bottlenecks and cannot cope with more complex attack methods. Also, implementing the DP in the VQ-VAE still needs more complete proof to confirm the correctness of applying the DP to VQ-VAE.

Future research could address these limitations by conducting more extensive experiments with various datasets and utilizing the FL to evaluate the performance in decentralized scenarios. In conclusion, our work represented a significant stride towards privacy-preserving image reconstruction and generation, providing a robust and efficient solution in the face of growing data privacy concerns. By advancing the SOTA in Privacy-preserving Artificial Intelligence (PPAI), we are optimistic that our work lays a strong foundation for future research and development in this rapidly evolving field.

4.7 Conclusion

In conclusion, this chapter presents a novel approach for privacy-preserving image generation by integrating the DP into the VQ-VAE framework, offering a valuable contribution to the field of privacy-preserving image reconstruction and generation. Our proposed DP-enabled VQ-VAE training flow RTF successfully and effectively balances privacy preservation and image generation performance, showcasing its potential to meet the growing demand for the PPAI solutions. Our experimental results reveal that the RTF reduces the privacy budget ϵ by approximately 34.80% on average across four diverse datasets, the MNIST, Fashion-MNIST, CIFAR-10,

and STL, while maintaining comparable performance to the VTF. This substantial reduction in privacy budget consumption underscores the robust privacy-preserving capabilities of our method when contrasted with the VTF. Despite the promising results achieved, there are still areas for further exploration and enhancement. Future research can focus on examining the scalability of the method to more extensive and more intricate datasets, optimizing the privacy-utility trade-offs through decentralized resolution such as the FL, and investigating the integration of other privacy-preserving mechanisms to bolster the method's privacy guarantees.

Chapter 5

Discussion

In this chapter, we make a report on the motivation, trad-offs, and future work of this study, which are described below.

5.1 Motivation

Data is collected through sensors, mobile devices, browsers, wearable devices, and more. This data uses Deep Learning (DL) applications across sectors like finance, e-commerce, social media, healthcare, and recommendation systems. As a result, an increasing number of data-driven DL applications are being designed, created, and optimized by major service providers, such as Apple, Google, and Amazon. These providers can be termed as the "data user." They harness vast quantities of data sourced from the individual who is the "data owner" to deliver tailored services. The public is aware that these services offer data owners both commercial and political benefits by enabling user recommendations, health monitoring, targeted advertising, and predictive insights. However, the data from these owners often contains sensitive or private information, raising concerns about privacy. Moreover, with the cloud-based DL services and computing converging to create powerful analytical platforms, it's crucial to prioritize privacy in the DL models on these platforms, especially when handling sensitive data. This thesis delves into privacy-preserving in the DL, outlining methods to ensure data confidentiality within the DL.

5.2 Trade-offs

5.2.1 Significance

When personal data is misused or used for unintended purposes, it can be exploited for unfair advantages. Combining large amounts of personal records with the DL algorithms introduces unpredictability, making it uncertain what insights might emerge or how much private information might be inadvertently disclosed. Thus, designing the DL algorithms that prioritize privacy-preserving applications is crucial. As for the consideration of privacy-preserving, there are three items that should be recognized.

I The method should ensure other data users cannot use the personal data for their own advantage without the data owner's permission.

II As thoughtful and intelligent human beings, everyone has private things that they do not want others to check or know, even if there are some legal rules or requirements that have been announced.

III Different places have varying degrees of privacy protection measures and regulations for different datasets. The same dataset, when used in different locations and subject to local laws and regulations, can produce significantly different results when modeled using various algorithms.

The core challenge lies in striking a balance between privacy and efficiency in DL applications, aiming to harness the data's potential while safeguarding individual privacy. Due to specific regulatory and application demands, we must not compromise privacy for the sake of enhanced utility. Furthermore, privacy measures should be methodically integrated rather than relying on random mechanisms.

According to Chapter 2, setting the encryption algorithm as the benchmark, we utilize an Efficient Integer Vector Homomorphic Encryption (EIVHE) scheme and propose Improved EIVHE (IEIVHE), implementing it in the DL to preserve users' privacy while using Deep Neural Network (DNN) model. To protect users' privacy

and security, a Fully Homomorphic Encryption (FHE) framework is used to encrypt the training and test dataset, which the DNN model is also adjusted to operate accordingly. Based on the characteristics of the HE algorithm, combined with efficient integral vectors, experiments are conducted on specific datasets. The algorithm is mainly applied to key-switching techniques to achieve internal ciphertext conversion and usage, ensuring that algebraic operations performed on ciphertext yield correct results. A comparative study was carried out focusing on parameters such as training set accuracy, test set accuracy, time overhead, and cycles. The results indicate that although the framework may slightly reduce accuracy, it ensures the privacy and security of users.

This technology can be applied to the DL on encrypted data, ensuring the confidentiality of training and inference while keeping the data used in an encrypted state. Even if the data is leaked, adversarial foes cannot decipher the actual meaning at the human level without the correct key, making privacy breaches infeasible. This prevents any damage to property, reputation, health, or honor. For example, in healthcare, where patient data can be highly sensitive. Homomorphic Encryption (HE) enables medical researchers and Artificial Intelligence (AI) models to develop more personalized medicine, diagnostics, and treatment plans by securely leveraging patient data without compromising privacy, which means that sensitive data can be analyzed and processed by third-party services without exposing the actual data, effectively protecting the privacy of the individuals or entities to which the data pertains.

Based on Channel-wise Homomorphic Encryption (CHE) proposed in Chapter 3, users who need to use cloud computing for secure and private computations can achieve faster and more convenient cloud data processing requirements. Moreover, this method tends to handle image data, so it is exceptionally advantageous for tasks related to image processing. For instance, performing secretive image recognition, image segmentation, and image generation using high-performance computers in the cloud. A similar application is in Adobe PhotoShop, where the cloud-based AI models are used to edit, crop, and transform images. Implementing more privacy-preserving methods would be a more legally compliant approach. Furthermore,

as it's well known, videos comprise stacked frames of images. This means that the method we propose can theoretically be applied to secretive video-processing tasks as well. Utilizing video-processing technology with privacy-preserving for government agencies will provide the public with safer and more secure surveillance and tracking for specific sensitive tasks.

Additionally, in practical applications, it's incredibly challenging to collect data with privacy concerns and sensitive information due to various factors. Data, at its essence, consists of facts readily transformed into computer-readable formats. Data collection is pervasive, from business analytics and engineering optimization to social science research and scientific studies. Each data set, with its unique patterns and characteristics, serves specific objectives. Take healthcare as an example: diverse imaging data, such as X-rays, Computed Tomography (CT) scans, and dermoscopy images, bolster the DL applications in diagnosing diseases and enhancing therapeutic methods.

Using the approach introduced in Chapter 4 we can implement more outsourced applications and allow the acquisition of valuable synthetic data from sensitive datasets through privacy-preserving means. For instance, most of the DL algorithms are required to infer directly from the feature vectors of the datasets rather than the statistics of the datasets; implementing Differential Privacy (DP) for privacy-preserving synthetic data generation is to provide a synthetic dataset that maintains the same statistical properties as the original dataset and has the same number of features and samples. Introducing the DP paves the way for further specialized application tasks, especially in healthcare, personalized education, and financial forecasting.

5.2.2 Limitation

To address privacy-preserving issues in the DL applications according to the approaches broadly grouped into categories: cryptographic-based and perturbation-based approaches; in this thesis, we propose two methods to process it, respectively.

In the study of Chapter 2, we discuss the EIVHE and IEIVHE algorithms to make the training and inference for ciphertext in the DNN. The algorithm used in

this chapter is, to some extent, capable of protecting user privacy while employing deep learning. In the experiments, although the initial tests did not show significant improvement, it is possible to consider more sophisticated schemes to enhance accuracy. However, due to the limited data and models in the experiments, the application as a whole is still somewhat biased. This is mainly reflected in the need to conduct experiments on more diversified datasets, such as structured data, and to test on more complex DNN models to seek more generalized results and comparisons. Since the HE requires the addition of noise during encryption, the final accuracy is definitely reduced, which is also a drawback of this scheme.

In the study of Chapter 3, we discuss the CHE method to make the inference over the ciphertext. This thesis only implements the Leveled Homomorphic Encryption (LHE) method instead of the Fully Homomorphic Encryption (FHE) method because of inference latency. The FHE is an encryption scheme that allows any number of functions on encrypted data without ever decrypting it. Instead of supporting an unlimited number of operations, the LHE supports a limited number of operations. The FHE should be able to evaluate any circuit; however, the LHE can evaluate circuits with a bounded depth, Multiplicative Depth (MD). Although the library tool of the LHE provided by Microsoft is helpful for implementing a cryptographic-based approach on the DL, it still needs to compute the MD in advance, which requires professional knowledge to operate and deduce. Furthermore, this thesis compares previous works according to the original paper's results since leaking source code of previous works and unable to record the exact results in the same situation, thus, we leave the direct comparison with the earlier studies under the same situation as a future work.

In the study of Chapter 4, we discuss the method of implementing a perturbation-based approach to make the synthetic image generation and sampling. This method is a DP-enabled method for the Vector Quantized-Variational AutoEncoder (VQ-VAE) model, striking a balance between privacy-preserving and performance. Unfortunately, this thesis only focuses on the centralized situation; in other words, it assumes process data in the curator. Furthermore, the method is an extraordinary approach for deploying the VQ-VAE model in the Gaussian mechanism DP, and we leave a

more comprehensive comparison of different models and different DP mechanisms. Also, implementing the DP in the VQ-VAE still needs more complete proof to confirm the correctness of applying the DP to VQ-VAE. The forward step should be aimed at the decentralized situation, also known as Federated Learning (FL).

5.3 Future Work

According to the recently published papers for the last 10 years, the methods of privacy-preserving in the DL can be roughly classed in five directions: the HE-based, Secure Multi-party Computation (SMPC)-based, the DP-based, Secure Enclaves (SE)-based, and Hybrid.

This thesis is only research on the two directions, the HE-based and DP-based, for privacy-preserving in the DL. There are still more charming approaches to deploying privacy-preserving in the DL. For instance, the SMPC-based method, the FL in privacy-preserving, is current hot-topic research, which is our next primary step for research in privacy-preserving concerning isolated data owners. Furthermore, the SE-based method is an approach quite related to hardware and physical machines, which can be set as the most secure approach in protecting the security and privacy of data. The Hybrid method can freely integrate specific technical methods based on application scenarios to achieve privacy-preserving for special requirements. Its advantage is that it does not rely on a single technical condition, allowing for the deployment of different privacy-preserving methods at various nodes based on actual requirements. However, this approach demands a relatively high level of knowledge and capability from the technical personnel and is generally used in environments with the highest privacy-preserving level.

In future work, judging from the two methods used in the thesis, firstly, the HE-based method, we will deal with the FHE scheme instead of the LHE, transform passive defense to active monitoring for privacy leakage and attack, and explore a more generic method for the DL; secondly, the DP-based method, we will follow up on input perturbation likes linear regression, algorithm perturbation which is implemented in thesis, objective perturbation likes logistic regression, and output

perturbation like naive bayes classification, respectively. For one step forward, we mainly focus on exploring the DP-based method in the FL situation. The three proposed methods of two targets are not entirely suitable for the complex network environment because of issues concerning the data collection mechanism from different parties while preserving privacy. Implementing an efficient, reasonable, and flexible privacy-preserving technique can improve the usability of more intelligent DL models for secure applications and sensitive information, which is the purpose of the "available but invisible."

Chapter 6

Conclusion

To address the privacy-preserving issues, the model should not reveal any private information about the training and inference data from the data owner and result owner, and meanwhile, the model also should not reveal any sensitive information about itself from the data user in Deep Learning (DL) models, the current doctoral thesis makes the systematical research on the privacy-preserving techniques, which relies heavily on the underlying data whose features include private or sensitive information.

In Chapter 2, we utilize an Efficient Integer Vector Homomorphic Encryption (EIVHE) and propose Improved EIVHE (IEIVHE) schemes in the DL. This chapter first outlines the HE algorithm based on efficient integer vectors, describing its definition and related fundamentals while also introducing the current state of research in the DL based on the HE algorithms. Then, inspired by the properties of efficient integer vectors and the characteristics of the HE algorithms, it proposes the application of the HE algorithms based on efficient integer vectors in Deep Neural Network (DNN) models. The main idea of this algorithm is first to encrypt the dataset using the HE algorithm with efficient integer vectors and then perform calculations using a DNN model. Experiments have proven that this method can protect user privacy and avoid the occurrence of privacy leaks by encrypting the dataset while using the DNN, which has improved accuracy 3.08% via the level of data from 86.42% to 89.05% on Modified National Institute of Standards and Technology (MNIST)

dataset with absolute value in the IEIVHE. Since this scheme applies cryptography to the DL, it needs to be adjusted according to the structure of the dataset before using other datasets. Because encrypted data is used in the DNN, attackers cannot understand the sensitive data obtained without the key, thereby ensuring the privacy and security of users. This scheme bridges cryptography and the DL and has more profound implications for future use.

In Chapter 3, we propose a Channel-wise Homomorphic Encryption (CHE) and Batch Normalization (BN) layer with Coefficient Merging (CM) to process ciphertext for inference, which is the cryptographic-based approach for preserving privacy. The CHE is a Leveled HE (LHE), which mainly provides the novel computation scheme "Onion" that makes HE rotation and mask operations on the vector of ciphertext. Furthermore, during the LHE, the MD is critical to the latency and computational complexity, and we propose two schemes, Convolution-Activation-BN (CAB) and Convolution-BN-Activation (CBA), improving the accuracy and decreasing the latency by reducing the MD logically; meanwhile, for validation of it, we also show the mathematical deduction and experimental test comparing the performance about two schemes. The results depict that the CHE realizes the highest accuracy of 99.32% and the shortest latency of 7.76 seconds on the grayscale MNIST dataset and an accuracy of 76.4% and a latency of 111.91 seconds on the color Canadian Institute for Advanced Research-10 classes (CIFAR-10) dataset in ciphertext inference.

In Chapter 4, we put forward integrating Differential Privacy-Stochastic Gradient Descent (DP-SGD) in the Tensorflow Privacy library into Vector Quantized-Variational AutoEncoder (VQ-VAE) model for privacy-preserving image generation and sampling, improving and speeding up the workflow with lower privacy budget, which is the perturbation-based approach for protecting privacy. In the DP, privacy budget ϵ and sensitivity are considerably related to the noise, affecting the privacy protection level. Significantly, the smaller the privacy budget ϵ, the more noise added, resulting in a higher level of privacy protection. We suggest our special approach for the VQ-VAE model by decreasing the privacy budget ϵ while presenting relevant performance. The results show that the proposed Revised Training Flow

(RTF) illustrates reducing privacy budget ϵ around 34.8% on average across four diverse test datasets.

The advantages of the current doctoral thesis are the following:

I It further strengthens and refines the research in the DL regarding the "available but invisible" of privacy-preserving, outlining the associated threats and proposing solutions for privacy-preserving.

II It bridges cryptography and the DL with an improved HE algorithm for training and test phases in the DNN model and has more profound implications for future utilization.

III It implements a novel algorithm to process data encrypted by the LHE in ciphertext inference and reduce latency for flexible applications.

IV It deploys an improved mechanism for the DP-enabled model, cutting down the privacy budget ϵ in image generation and sampling.

To conclude, this thesis has made significant contributions to the emerging field of Privacy-preserving Deep Learning (PPDL). This progress is rooted in the application of cryptographic knowledge and perturbation techniques. This research represents a fundamental step in addressing challenges within the PPDL domain. By outlining the design of the framework, this work provides a roadmap for future explorations in PPDL. It aims to foster the development of necessary measures to protect private and sensitive data in data-driven models, ensuring that privacy is not compromised. It is hoped that this study will serve as a catalyst for various advancements in the PPDL, particularly in the area of discrete data application scenarios.

Acknowledgement

First and foremost, I wish to express my profound gratitude to my advisor, Professor Tatsuya Mori, for his unwavering support, invaluable guidance, and insightful feedback throughout my research. His expertise and mentorship have been pivotal to my academic journey. Under Professor Mori's guidance, I undertook to broaden my research studies, learning and grasping advanced academic knowledge and skills in this research field. I particularly learned how to conduct research and became a qualified, dedicated, and responsible researcher. The professor is like a life mentor to me, providing immense academic support and taking meticulous care of me daily. From the earliest stages of conceptualizing this research to the final moments of writing, Professor Mori's wisdom, patience, and experience have been my guiding lights. His attention to detail, commitment to academic rigor, and passion for Network Security and Privacy not only shaped this thesis but also transformed my approach to research and critical thinking.

I would also like to thank my co-advisor and co-writer, Professor Yamana, for the invaluable suggestions he provided regarding my research in the field of privacy-preserving. He gave me a renewed and unique perspective on this research area. Through numerous online and offline discussions and email exchanges, I have gained a lot of critical feedback and constructive suggestions, significantly enriching my research work.

Special appreciation goes to the members of Mori's Network Security Laboratory for the stimulating discussions, collaborative spirit, and countless hours spent in the laboratory. Their camaraderie made the challenging days more accessible and the successes sweeter.

I am deeply thankful to the China Scholarship Council (CSC), which provided

the financial support that made my research possible. I would also like to thank Professor Yamana for allowing me to participate in the CREST project of the Japan Science and Technology Agency (JST) and thank Professor Mori for supporting me in working as a research assistant at the Faculty of Science and Engineering, Waseda University.

On a personal note, I owe my deepest gratitude to my family for their unconditional love, encouragement, and belief in me. To my friends, thank you for being my pillars of support and for the moments of laughter amidst the stresses.

Next, I would like to acknowledge the countless researchers and authors in the field of privacy-preserving whose works have inspired and paved the way for this research. I am privileged to contribute to this body of knowledge.

Lastly, I want to express my acknowledgment and best wishes to everyone who has helped me, ending with one motto:

Less interests, More interest.

Bibliography

[1] Mohammad Al-Rubaie and J. Morris Chang. Privacy-preserving machine learning: Threats and solutions. *IEEE Secur. Priv.*, Vol. 17, No. 2, pp. 49–58, 2019.

[2] J Morris Chang, Di Zhuang, G Samaraweera, and G Dumindu Samaraweera. *Privacy-Preserving Machine Learning*. Simon and Schuster, 2023.

[3] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pp. 1310–1321. ACM, 2015.

[4] Kwangjo Kim and Harry Chandra Tanuwidjaja. *Privacy-Preserving Deep Learning - A Comprehensive Survey*. Springer, 2021.

[5] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, Vol. 4, No. 11, pp. 169–180, 1978.

[6] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, Vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 201–210. JMLR.org, 2016.

[7] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. Privacy-preserving classification on deep neural network. *IACR Cryptol. ePrint Arch.*, p. 35, 2017.

[8] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Deep neural networks classification over encrypted data. In Gail-Joon Ahn, Bhavani Thuraisingham, Murat Kantarcioglu, and Ram Krishnan, editors, *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy, CODASPY 2019, Richardson, TX, USA, March 25-27, 2019*, pp. 97–108. ACM, 2019.

[9] Takumi Ishiyama, Takuya Suzuki, and Hayato Yamana. Highly accurate CNN inference using approximate activation functions over homomorphic encryption. In Xintao Wu, Chris Jermaine, Li Xiong, Xiaohua Hu, Olivera Kotevska, Siyuan Lu, Weija Xu, Srinivas Aluru, Chengxiang Zhai, Eyhab Al-Masri, Zhiyuan Chen, and Jeff Saltz, editors, *2020 IEEE International Conference on Big Data (IEEE BigData 2020), Atlanta, GA, USA, December 10-13, 2020*, pp. 3989–3995. IEEE, 2020.

[10] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption. *CoRR*, Vol. abs/2104.03152, , 2021.

[11] Ahmad Al Badawi, Chao Jin, Jie Lin, Chan Fook Mun, Sim Jun Jie, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, and Vijay Ramaseshan Chandrasekhar. Towards the alexnet moment for homomorphic encryption: Hcnn, the first homomorphic CNN on encrypted data with gpus. *IEEE Trans. Emerg. Top. Comput.*, Vol. 9, No. 3, pp. 1330–1343, 2021.

[12] Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, and Jong-Seon No. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, Vol. 10, pp. 30039–30054, 2022.

[13] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *CoRR*, Vol. abs/1711.05189, , 2017.

[14] Edward Chou, Josh Beal, Daniel Levy, Serena Yeung, Albert Haque, and Li Fei-Fei. Faster cryptonets: Leveraging sparsity for real-world encrypted inference. *CoRR*, Vol. abs/1811.09953, , 2018.

[15] Runhua Xu, James B. D. Joshi, and Chao Li. Cryptonn: Training neural networks over encrypted data. In *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019*, pp. 1199–1209. IEEE, 2019.

[16] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[17] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *CoRR*, Vol. abs/1610.05492, , 2016.

[18] Lita Yang and Boris Murmann. Approximate sram for energy-efficient, privacy-preserving convolutional neural networks. In *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 689–694. IEEE, 2017.

[19] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. DP-CGAN: differentially private synthetic data and label generation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 98–104. Computer Vision Foundation / IEEE, 2019.

[20] Bangzhou Xin, Wei Yang, Yangyang Geng, Sheng Chen, Shaowei Wang, and Liusheng Huang. Private FL-GAN: differential privacy synthetic data generation based on federated learning. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pp. 2927–2931. IEEE, 2020.

[21] Dihong Jiang, Guojun Zhang, Mahdi Karami, Xi Chen, Yunfeng Shao, and Yaoliang Yu. Dp^2-vae: Differentially private pre-trained variational autoencoders. *CoRR*, Vol. abs/2208.03409, , 2022.

[22] PLC INTEL. Intel software guard extensions programming reference. *Accessed on*, Vol. 8, No. 2022, pp. 329298–002, 2014.

[23] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett

Witchel. Chiron: Privacy-preserving machine learning as a service. *CoRR*, Vol. abs/1803.05961, , 2018.

[24] Florian Tramèr and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[25] Andrew Law, Chester Leung, Rishabh Poddar, Raluca Ada Popa, Chenyu Shi, Octavian Sima, Chaofan Yu, Xingmeng Zhang, and Wenting Zheng. Secure collaborative training and inference for xgboost. In Benyu Zhang, Raluca Ada Popa, Matei Zaharia, Guofei Gu, and Shouling Ji, editors, *PPMLP'20: Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice, Virtual Event, USA, November, 2020*, pp. 21–26. ACM, 2020.

[26] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha P. Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pp. 1651–1669. USENIX Association, 2018.

[27] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow: Secure tensorflow inference. *CoRR*, Vol. abs/1909.07814, , 2019.

[28] Théo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning. *CoRR*, Vol. abs/1811.04017, , 2018.

[29] Google Inc. Tensorflow privacy 0.8.7, 2022. `https://www.tensorflow.org/responsible_ai/privacy/guide`.

[30] Changzhi Luo, Zhetao Li, Kaizhu Huang, Jiashi Feng, and Meng Wang. Zero-shot learning via attribute regression and class prototype rectification. *IEEE Trans. Image Process.*, Vol. 27, No. 2, pp. 637–648, 2018.

[31] Guosheng Hu, Xiaojiang Peng, Yongxin Yang, Timothy M. Hospedales, and Jakob Verbeek. Frankenstein: Learning deep face representations using small data. *IEEE Trans. Image Process.*, Vol. 27, No. 1, pp. 293–303, 2018.

[32] Hongchao Zhou and Gregory W. Wornell. Efficient homomorphic encryption on integer vectors and its applications. In *2014 Information Theory and Applications Workshop, ITA 2014, San Diego, CA, USA, February 9-14, 2014*, pp. 1–9. IEEE, 2014.

[33] Joppe W. Bos, Kristin E. Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, Vol. 8308 of *Lecture Notes in Computer Science*, pp. 45–64. Springer, 2013.

[34] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, p. 144, 2012.

[35] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pp. 309–325. ACM, 2012.

[36] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.*, Vol. 43, No. 2, pp. 831–871, 2014.

[37] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in lwe-based homomorphic encryption. In *Public-Key Cryptography–PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26–March 1, 2013. Proceedings 16*, pp. 1–13. Springer, 2013.

[38] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pp. 169–178. ACM, 2009.

[39] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, Vol. 26, No. 1, pp. 96–99, 1983.

[40] Qingchen Zhang, Laurence T. Yang, and Zhikui Chen. Privacy preserving

deep computation model on cloud for big data feature learning. *IEEE Trans. Computers*, Vol. 65, No. 5, pp. 1351–1362, 2016.

[41] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, Vol. 3378 of *Lecture Notes in Computer Science*, pp. 325–341. Springer, 2005.

[42] Qingchen Zhang, Laurence T. Yang, Zhikui Chen, Peng Li, and M. Jamal Deen. Privacy-preserving double-projection deep computation model with crowdsourcing on cloud for big data feature learning. *IEEE Internet Things J.*, Vol. 5, No. 4, pp. 2896–2903, 2018.

[43] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.

[44] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pp. 308–318. ACM, 2016.

[45] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, Vol. 4. Springer, 2006.

[46] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[47] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, Vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015.

[48] Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri,

Kleomenis Katevas, Hamid R Rabiee, Nicholas D Lane, and Hamed Haddadi. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal*, Vol. 7, No. 5, pp. 4505–4518, 2020.

[49] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security*, Vol. 13, No. 5, pp. 1333–1345, 2017.

[50] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pp. 896–902. IEEE, 2015.

[51] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits, 2010.

[52] Tianying Xie, Hayato Yamana, and Tatsuya Mori. CHE: channel-wise homomorphic encryption for ciphertext inference in convolutional neural network. *IEEE Access*, Vol. 10, pp. 107446–107458, 2022.

[53] Manjusha Deshmukh, Udhav Bhosale, et al. Image fusion and image quality assessment of fused images. *International Journal of Image Processing (IJIP)*, Vol. 4, No. 5, p. 484, 2010.

[54] Khalid Muhammad, Kiki Ariyanti Sugeng, and Hendri Murfi. Machine learning with partially homomorphic encrypted data. In *Journal of Physics: Conference Series*, Vol. 1108, p. 012112. IOP Publishing, 2018.

[55] SIMD (single instruction multiple data processing). In Borko Furht, editor, *Encyclopedia of Multimedia, 2nd Ed*, pp. 817–819. Springer, 2008.

[56] Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Ramy Masalha, Guy Moshkowich, Dov Murik, Hayim Shaul, and Omri Soceanu. Helayers: A tile tensors framework for large neural networks on encrypted data. Vol. 2023, pp. 325–342, 2023.

[57] Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin E. Lauter, Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. CHET: an optimizing compiler for fully-homomorphic neural-network inferencing. In Kathryn S. McKinley and Kathleen Fisher, editors, *Proceedings of the 40th*

ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019, pp. 142–156. ACM, 2019.

[58] Qian Lou and Lei Jiang. HEMET: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Vol. 139 of *Proceedings of Machine Learning Research*, pp. 7102–7110. PMLR, 2021.

[59] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, Vol. 79, No. 8, pp. 2554–2558, 1982.

[60] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, Vol. abs/1803.08375, , 2018.

[61] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Trans. Neural Networks Learn. Syst.*, Vol. 33, No. 12, pp. 6999–7019, 2022.

[62] Krizhevsky Alex, Hinton Geoffrey, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Canada, 2009.

[63] Pascal Aubry, Sergiu Carpov, and Renaud Sirdey. Faster homomorphic encryption is not enough: Improved heuristic for multiplicative depth minimization of boolean circuits. In Stanislaw Jarecki, editor, *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*, Vol. 12006 of *Lecture Notes in Computer Science*, pp. 345–363. Springer, 2020.

[64] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, Vol. 10624 of *Lecture Notes in Computer Science*, pp. 409–437.

Springer, 2017.

[65] Harvey L. Garner. The residue number system. *IRE Trans. Electron. Comput.*, Vol. 8, No. 2, pp. 140–147, 1959.

[66] Qiao Zhang, Cong Wang, Hongyi Wu, Chunsheng Xin, and Tran V. Phuong. Gelu-net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 3933–3939. ijcai.org, 2018.

[67] Peichen Xie, Bingzhe Wu, and Guangyu Sun. BAYHENN: combining bayesian deep learning and homomorphic encryption for secure DNN inference. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 4831–4837. ijcai.org, 2019.

[68] Tianying Xie and Yantao Li. Efficient integer vector homomorphic encryption using deep learning for neural networks. In Long Cheng, Andrew Chi-Sing Leung, and Seiichi Ozawa, editors, *Neural Information Processing - 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I*, Vol. 11301 of *Lecture Notes in Computer Science*, pp. 83–95. Springer, 2018.

[69] Junghyun Lee, Eunsang Lee, Joon-Woo Lee, Yongjune Kim, Young-Sik Kim, and Jong-Seon No. Precise approximation of convolutional neural networks for homomorphically encrypted data. *CoRR*, Vol. abs/2105.10879, , 2021.

[70] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pp. 707–721. ACM, 2018.

[71] Jun Liu, Yuan Tian, Yu Zhou, Yang Xiao, and Nirwan Ansari. Privacy

preserving distributed data mining based on secure multi-party computation. *Comput. Commun.*, Vol. 153, pp. 208–216, 2020.

[72] Paverd Andrew, Martin Andrew, and Brown Ian. Modelling and automatically analysing privacy properties for honest-but-curious adversaries. Technical report, University of Oxford, Oxford, UK, 2014.

[73] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pp. 162–167. IEEE Computer Society, 1986.

[74] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, Vol. 4052 of *Lecture Notes in Computer Science*, pp. 1–12. Springer, 2006.

[75] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious multi-party machine learning on trusted processors. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pp. 619–636. USENIX Association, 2016.

[76] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 19–38. IEEE Computer Society, 2017.

[77] Weibei Fan, Jing He, Mengjiao Guo, Peng Li, Zhijie Han, and Ruchuan Wang. Privacy preserving classification on local differential privacy in data centers. *J. Parallel Distributed Comput.*, Vol. 135, pp. 70–82, 2020.

[78] Yu Chen, Fang Luo, Tong Li, Tao Xiang, Zheli Liu, and Jin Li. A training-integrity privacy-preserving federated learning scheme with trusted execution environment. *Inf. Sci.*, Vol. 522, pp. 69–79, 2020.

[79] Ramy E. Ali, Jinhyun So, and Amir Salman Avestimehr. On polynomial approximations for privacy-preserving and verifiable relu networks. *CoRR*,

Vol. abs/2011.05530, , 2020.

[80] Shai Halevi and Victor Shoup. Algorithms in helib. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, Vol. 8616 of *Lecture Notes in Computer Science*, pp. 554–571. Springer, 2014.

[81] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

[82] Microsoft SEAL (release 3.7). `https://github.com/Microsoft/SEAL`, September 2021. Microsoft Research, Redmond, WA.

[83] Python Software Foundation. Python 3.9.0, 2020. `https://www.python.org/downloads/release/python-390/`.

[84] Huelse. Seal-python, 2021.

[85] Fenghua Li, Hui Li, Ben Niu, and Jinjun Chen. Privacy computing: Concept, computing framework and future development trends. *IACR Cryptol. ePrint Arch.*, p. 1145, 2018.

[86] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, Vol. 63, No. 11, pp. 139–144, 2020.

[87] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[88] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman

Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6306–6315, 2017.

[89] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in neural information processing systems*, Vol. 20, , 2007.

[90] Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

[91] Jungang Xu, Hui Li, and Shilong Zhou. An overview of deep generative models. *IETE Technical Review*, Vol. 32, No. 2, pp. 131–139, 2015.

[92] Hirotugu Akaike. Fitting autoregreesive models for prediction. In *Selected Papers of Hirotugu Akaike*, pp. 131–135. Springer, 1969.

[93] Eberhard Engel and Reiner M Dreizler. From explicit to implicit density functionals. *Journal of computational chemistry*, Vol. 20, No. 1, pp. 31–50, 1999.

[94] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

[95] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 1175–1191. ACM, 2017.

[96] NhatHai Phan, Yue Wang, Xintao Wu, and Dejing Dou. Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 1309–1316. AAAI Press, 2016.

[97] Tsubasa Takahashi, Shun Takagi, Hajime Ono, and Tatsuya Komatsu. Differentially private variational autoencoders with term-wise gradient aggregation.

CoRR, Vol. abs/2006.11204, , 2020.

[98] Benjamin Weggenmann, Valentin Rublack, Michael Andrejczuk, Justus Mattern, and Florian Kerschbaum. DP-VAE: human-readable text anonymization for online reviews with differentially private variational autoencoders. In Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini, editors, *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pp. 721–731. ACM, 2022.

[99] Hisaichi Shibata, Shouhei Hanaoka, Yang Cao, Masatoshi Yoshikawa, Tomomi Takenaga, Yukihiro Nomura, Naoto Hayashi, and Osamu Abe. Local differential privacy image generation using flow-based deep generative models. *CoRR*, Vol. abs/2212.10688, , 2022.

[100] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, p. 125. ISCA, 2016.

[101] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14837–14847, 2019.

[102] Surajit Chaudhuri, Umeshwar Dayal, and Vivek R. Narasayya. An overview of business intelligence technology. *Commun. ACM*, Vol. 54, No. 8, pp. 88–98, 2011.

[103] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. Generative models for effective ML on private, decentralized datasets. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[104] Gregor Schram, Rui Wang, and Kaitai Liang. Using autoencoders on differentially private federated learning gans. *CoRR*, Vol. abs/2206.12270, , 2022.

[105] Alberto Blanco-Justicia, David Sánchez, Josep Domingo-Ferrer, and Krishnamurty Muralidhar. A critical review on the use (and misuse) of differential privacy in machine learning. *ACM Comput. Surv.*, Vol. 55, No. 8, pp. 160:1–160:16, 2023.

[106] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, Vol. abs/1606.03498, , 2016.

[107] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6626–6637, 2017.

[108] Shahram Shirani. Data compression: The complete reference (by d. salomon; 2007) [book review]. *IEEE Signal Process. Mag.*, Vol. 25, No. 1, pp. 147–149, 2008.

[109] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, Vol. 9, No. 3-4, pp. 211–407, 2014.

[110] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[111] M Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, Vol. 22, No. 1, pp. 1–72, 1906.

[112] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new

look at signal fidelity measures. *IEEE signal processing magazine*, Vol. 26, No. 1, pp. 98–117, 2009.

[113] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, Vol. 7, No. 3, pp. 17–51, 2016.

[114] Google Inc. Tensorflow 2.10, 2022. `https://www.tensorflow.org/`.

[115] Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.*, Vol. 29, No. 6, pp. 141–142, 2012.

[116] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, Vol. abs/1708.07747, , 2017.

[117] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[118] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, Vol. 15 of *JMLR Proceedings*, pp. 215–223. JMLR.org, 2011.

List of Research Achievements

Journal Papers (Reviewed)

1. Tianying Xie, Hayato Yamana, Tatsuya Mori, "CHE: Channel-Wise Homomorphic Encryption for Ciphertext Inference in Convolutional Neural Network," IEEE Access, Page 107446 - 107458, Volume 10, September 2022 (online access: https://ieeexplore.ieee.org/document/9903645).
2. Tianying Xie, Yantao Li, "Adding Gaussian Noise to DeepFool for Robustness based on Perturbation Directionality," Australian Journal of Intelligent Information Processing Systems, Page 44-54, Volume 16, Issue 3, 2019 (online access: http://ajiips.com.au/papers/V16.3/v16n3_46-55).

Conference Papers (Reviewed)

1. Tianying Xie, Yantao Li, "A Gradient-Based Algorithm to Deceive Deep Neural Networks," Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, Part of the Communications in Computer and Information Science book series (CCIS), Springer, Page 57-65, Volume 1142, December 2019 (online access: https://link.springer.com/chapter/10.1007/978-3-030-36808-1_7).
2. Tianying Xie, Yantao Li, "Efficient Integer Vector Homomorphic Encryption Using Deep Learning for Neural Networks," Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, Part of the Lecture Notes in Computer Science book series (LNTCS), Springer, Page 83-95, Volume 11301, December 2018 (online access: https://link.springer.com/chapter/10.1007/978-3-030-04167-0_8).

Invited Talks (Oral Presentation)

1. Tianying Xie, “Efficient Integer Vector Homomorphic Encryption Using Deep Learning for Neural Networks,” The WUHU workshop 2020, 25 November 2020.

Others

1. Tianying Xie, Hayato Yamana, Tatsuya Mori, “Improving the Performance of Deep Learning Image Classification Applied with Homomorphic Encryption,” / “準同型暗号を活用した画像分類深層学習の性能向上にむけて,” コンピュータセキュリティシンポジウム 2023（CSS2023）論文集, pp. 63-70 2023 年 11 月
2. Tianying Xie, Hayato Yamana, Tatsuya Mori, “Can We Speed Up Secure Deep Learning Algorithms on Fully Homomorphic Encryption Techniques?” 早慶合同　博士キャリアイベント 2022

Awards

1. Tianying Xie, Hayato Yamana, Tatsuya Mori, コンピュータセキュリティシンポジウム 2023（CSS2023）CSS 学生論文賞 “Improving the Performance of Deep Learning Image Classification Applied with Homomorphic Encryption,” 2023 年 11 月
2. Tianying Xie, Hayato Yamana, Tatsuya Mori, コンピュータセキュリティシンポジウム 2023（CSS2023）第 9 回プライバシーワークショップ (PWS2023) 学生論文賞 “Improving the Performance of Deep Learning Image Classification Applied with Homomorphic Encryption,” 2023 年 11 月

Copyrights