Hindawi
Mobile Information Systems
Volume 2021, Article ID 5510857, 11 pages
https://doi.org/10.1155/2021/5510857

Research Article

Deep Learning Data Privacy Protection Based on Homomorphic Encryption in AIoT

Yichuan Wang, Xiaolong Liang, Xinhong Hei, Wenjiang Ji, and Lei Zhu

College of Computer Science and Engineering, Xi'an University of Technology, 710048 Xi'an, China

Correspondence should be addressed to Xinhong Hei; heixinhong@xaut.edu.cn

Received 19 February 2021; Revised 28 April 2021; Accepted 12 June 2021; Published 25 June 2021

Academic Editor: Xiaohong Jiang

With the rapid development of 5G technology, its high bandwidth, high reliability, low delay, and large connection characteristics have opened up a broader application field of IoT. Moreover, AIoT (Artificial Intelligence Internet of Things) has become the new development direction of IoT. Through deep learning of real-time data provided by the Internet of Things, AI can judge user habits more accurately, make devices behave in line with user expectations, and become more intelligent, thus improving product user experience. However, in the process, there is a lot of data interaction between the edge and the cloud. Given that the shared data contain a large amount of private information, preserving information security on the shared data is an important issue that cannot be neglected. In this paper, we combine deep learning with homomorphic encryption algorithm and design a deep learning network model based on secure multiparty computing (MPC). In the whole process, we realize that the cloud only owns the encryption samples of users, and users do not own any parameters or structural information related to the model. In the experimental part, we input the encrypted Mnist and Cifar-10 datasets into the model for testing, and the results show that the classification accuracy rate of the encrypted Mnist can reach 99.21%, which is very close to the result under plaintext. The classification accuracy rate of encrypted Cifar-10 can reach 91.35%, slightly lower than the test result in plaintext and better than the existing deep learning network model that can realize data privacy protection.

1. Introduction

With the rapid development of 5G technology, 5G is leading the evolution of IoT standard [1–3]. However, it should also be noted that IoT standard mainly solves the problem of data transmission technology [4, 5], while AIoT focuses on the new application form of IoT, emphasizing services, especially back-end processing and application oriented to the Internet of Things [6, 7]. The massive and complex data generated by the Internet of Things need to be analyzed and processed, and AI technology is exactly the best choice for effective information processing. It can make intelligent products better understand the user's intention [8, 9]. AI's data can only be continuously provided by IoT. The massive data provided by IoT can enable AI to acquire knowledge quickly. Integration with AI technology can bring broader market prospects for the Internet of Things. In particular, deep learning is an important subfield of AI. Its motivation is to build neural networks that simulate the human brain for analytical learning [10], and it imitates the mechanism of human brain to interpret data, such as images, sounds, and texts.

Neural network technology was originated in the 1950s and 1960s as perceptron, with an input layer, an output layer, and a hidden layer [11, 12]. The input eigenvectors reach the output layer through the transformation of the hidden layer, and the classification results are obtained at the output layer. The early perceptrons were driven by Rosenblatt. However, the most serious problem with Rosenblatt's single-layer perceptron is that it is incapable of handling slightly more complex functions (such as the most typical "xor" operations). With the development of mathematics, this shortcoming was not overcome until the 1980s by the invention of multilayer perceptron by Rumelhart, Williams, Hinton, LeCun, and others. A multilayer perceptron, as the name

suggests, is a perceptron with multiple hidden layers, as shown in Figure 1.

Multilayer perceptrons can get rid of the constraints of early discrete transmission functions and use continuous functions such as Sigmoid or tanh to simulate the response of neurons to excitation. In terms of training algorithms, Werbos's back-propagation (BP) algorithm is used. This is what we now call a neural network.

According to the report, a series of companies, including Google, Facebook, Baidu, and Alibaba, have also publicly announced that artificial intelligence will be their next strategic focus. Based on the technology backgrounds of these tech giants and their huge investments in deep learning, deep learning is now making extraordinary progress. Moreover, with the rapid development of cloud service mode, these enterprises deploy the corresponding deep learning model in the cloud as the computing vendors to provide services for users according to the different solutions. On the one hand, it provides users with powerful services. On the other hand, after uploading their data, such as photos, voice, and video, to the cloud computing, users will face the risk of data privacy disclosure because they cannot control their subsequent usage. In addition, we know that the neural network model will get better with the increase and diversification of the training dataset. However, in areas such as finance and health care, laws or regulations do not allow the sharing of personal data, so we may not be able to get a good deep learning model based on the limited data [13]. With modern society paying more and more attention to privacy protection, how to ensure the privacy of data and the effectiveness of algorithms in the process of computing has become a major problem in the field of machine learning.

The appearance of homomorphic encryption makes the application of ciphertext in the field of deep learning possible. Rivest was the first to put forward the concept of homomorphic encryption in 1978 [14]. Homomorphic encryption is a form of encryption, which allows people to perform specific algebraic operations on ciphertext to obtain the result of encryption. Decrypting the ciphertext will obtain the same result as performing the same operation on the plaintext. There are no real homomorphic encryption algorithms available, and most of the existing ones are additive only, such as Benaloh [15], as well as RSA [16] and EIGamal algorithms that support multiplicative homomorphisms. In September 2009, IBM researcher Craig Gentry published a paper [17] proposing a homomorphic encryption algorithm based on ideal lattice, which became a solution that could realize all attributes of homomorphic encryption. This is a milestone in homomorphic cryptography, but it cannot be applied in practice due to the high cost of the scheme itself, computational model, and high security. After that scholars have proposed a degree of complete homomorphic encryption (SWHE) [18]. This kind of encryption method is only applicable to operations based on low-order polynomials.

Although the theory of homomorphic encryption can be arbitrary calculation, but in practice, we should pay attention to the following characteristics of the encryption: only support integer data; the depth of multiplication needs to be fixed, so the addition and multiplication cannot be carried out indefinitely. And homomorphic encryption does not support operations such as comparison and maximization. Therefore, homomorphic encryption cannot be directly applied to deep learning.

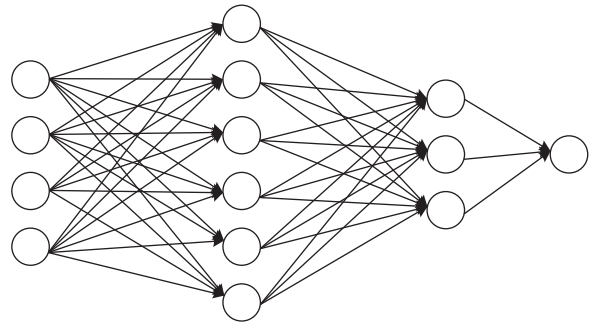

Figure 1: Perceptron with multiple hidden layers.

In the following paragraphs, we will discuss these problems and then propose solutions. And our contributions are as follows:

(i) By combining deep learning network with the homomorphic encryption algorithm, we designed an architecture based on secure multiparty computing, that is, a series of distributed processing such as standardization and encryption of user privacy data at the edge, then reasoning based on cloud deep learning application, and finally returning the results to the edge for decryption. Thus, the privacy protection of user data is realized. In the whole process, we realized that the cloud only owns the user's encryption sample, and the user does not own any parameters or structural information related to the model.

(ii) We designed the corresponding CNN model and encrypted Mnist and Cifar-10, respectively, and then tested as the dataset. The results showed that the classification accuracy rate of the encrypted Minst dataset can reach 99.21%, which is very close to the test result under plaintext and also is close to the accuracy of state-of-art model. The classification accuracy rate of Cifar-10 encrypted dataset can reach 91.35%, slightly lower than the test result in plaintext and better than the existing deep learning network model that can realize user data privacy protection. And the unit sample takes an average of only 11 seconds to complete an inference in the cloud without parallel optimization.

The rest of the paper is organized as follows.

In Section 2, we mainly introduce the related work. In Section 3, our model will be introduced. In Section 4, the method of sample pretreatment and inference in the cloud is given through experiments, and finally the experimental comparison results with other relevant models are given.

2. Related Work

There have been several excellent contributions in this field, such as follows.

Graepel et al. used a somewhat HE scheme to train two machine learning classifiers: linear mean and Fisher's linear discriminate (FLD). They proposed division-free algorithms to adapt to limitations of HE algorithms. They focussed on simple classifiers such as the linear means classifier and did not consider more complex algorithms.

Bost et al. [19] used a combination of three homomorphic systems (Quadratic Residuosity, Paillier, and BGV schemes) and garbled circuits to provide privacy-preserving classification for three different machine learning algorithms, namely, Hyperplane Decision, Naive Bayes, and Decision Trees. Their approach considers only classical machine learning algorithms and is only efficient for small datasets.

Dowlin et al. [20] proposed CryptoNets. They used the homomorphic encryption scheme of Bos et al. [19], which is very closely related to the schemes in López-Alt et al. [21] and Stehlé & Steinfeld [22]. This scheme is a leveled homomorphic encryption scheme, which allows adding and multiplying encrypted messages but requires that one knows in advance the complexity of the arithmetic circuit that is to be applied to the data. And they used the square function as the active function. However, the square function results in that the computing resource cost becomes prohibitive as the number of layers increases, which is not adapted deep learning.

There are also a few recent work that look at privacy issues in training phase, specifically for the back-propagation algorithm. Bu et al. proposed a privacy-preserving back-propagation algorithm based on BGV encryption scheme on cloud. Their proposed algorithm offloads the expensive operations to the cloud and uses BGV to protect the privacy of the data during the learning process. Zhang et al. also proposed using BGV encryption scheme to support the secure computation of the high-order back-propagation algorithm efficiently for deep computation model training on cloud. In their approach, to avoid a multiplicative depth too big, after each iteration, the updated weights are sent to the parties to be decrypted and reencrypted. Thus, the communication complexity of the solution is very high.

As we know, privacy data processing is based on two considerations. One is to obtain the calculation result you want on the premise of not exposing your data to the other party. The other is that when the attacker breaches the protection, the obtained data or intermediate data are still meaningless ciphertext. Thus, we design the privacy of data using the neural network classification, and it can be described as follows: the two parties are Alice and Bob; Alice is data hold party; Bob is neural network holds a party. Alice want to be on the premise of not revealing information about yourself, to get what you want to use Bob's neural network classification results. After Alice, to get the results, Bob does not know Alice's data; Alice also does not know about Bob's neural network.

Further description is as follows: first of all, we need to make it clear that there is already a trained model which is using plaintext in the cloud; our work is mainly focused on the stage of the reasoning model; according to the complexity of the dataset, we design different convolution neural network structures; network structure can learn more rich and more complex high-dimensional feature with better performance and better adaptability. In the reasoning stage, we present an interaction model based on secure multiparty computing, and the optimized Paillier encryption algorithm can be used to protect users' privacy data and obtain the expected reasoning results.

3. The Model

In this section, firstly, we introduce the Paillier algorithm and how to optimize it to operate on real numbers. And then, we analyze the characteristics of the convolutional neural network and each layer of the network to combine it with the homomorphic encryption algorithm. Finally, the data interaction architecture based on MPC will be introduced.

3.1. Paillier Encryption. In essence, homomorphic encryption refers to such an encryption function, which encrypts the plaintext by adding and multiplying operations on the ring and then encrypts the ciphertext after encryption, and the result is equivalent. Because of this good nature, people can entrust a third party to process the data without revealing the information. Below, we introduce the working steps and properties of the Paillier algorithm.

3.1.1. Key Generation. Take two large prime numbers p and q at random, and calculate their product N and the least common multiple λ of $p-1$ and $q-1$:

$$\begin{aligned} N &= p \cdot q, \\ \lambda &= lcm(p-1, q-1). \end{aligned} \tag{1}$$

An integer $g \in Z^*_{N^2}$ is randomly selected, $Z^*_{N^2}$ represents the set of all Numbers in Z_{N^2} that are interprime with N^2, and Z_{N^2} is the integer less than N^2, where λ is the private key and (N, g) is the public key.

3.1.2. The Encryption Process. If $m \in Z_N$ is the ciphertext to be encrypted, the encryption algorithm is as follows:

$$c = E[m, r] = g^m r^N \mathrm{mod} N^2. \tag{2}$$

Among them, $m \in Z^*_{N^2}$ is a randomly selected integer and c is the encrypted data. Therefore, for the same plaintext m, different ciphertext c will be generated due to the different r selected, but the same m can be restored after decryption.

3.1.3. Decryption Process. If $c \in Z^*_{N^2}$ is the ciphertext to be decrypted, the decryption algorithm is as follows:

$$D(c) = \frac{L\left(c^\lambda \mathrm{mod} N^2\right)}{L\left(g^\lambda \mathrm{mod} N^2\right)} \mathrm{mod} N, \tag{3}$$

where $L(*)$ is defined as follows:

$$L(u) = \frac{u-1}{N}. \tag{4}$$

The homomorphism of Paillier is shown as follows:

$$D\left[E[m_1, r_1]E[m_2, r_2] \bmod N^2\right] = m_1 + m_2 \bmod N,$$
$$D\left[E[m_1, r_1]^k \bmod N^2\right] = m * k \bmod N, k \in Z_N. \tag{5}$$

3.1.4. Processing of Real Numbers. Paillier encryption is only defined for nonnegative integers less than N. Since we frequently want to use signed integers and floating-point numbers, values should be encoded as a valid integer before encryption. The preprocessing of negative and floating-point numbers is as follows:

(i) Representing signed integers is relatively easy. We exploit the modular arithmetic properties of the Paillier scheme. We choose to represent only integers between [−max_int, max_int], where the max_int approximately equals $N/3$ (larger integers may be treated as floats). The range of values between [max_int, N-max_int] is reserved for detecting overflows. Schematic diagram of numerical range is shown in Figure 2.

We use $x(l)$ to represent the input value and $z(l)$ to represent the output value, so the above process can be expressed as the following formula:

$$x(l) \bmod N = z(l) = \begin{cases} x(l), & x(l) \geq 0, \\ x(l) + N, & \text{other}. \end{cases} \tag{6}$$

(ii) Representing floating-point numbers as integers is a harder task. Here, we use a variant of fixed-precision arithmetic. In fixed precision, we can encode by multiplying every float by a large number (e.g., 1e6) and rounding the resulting product. And we can decode by dividing by that number. As shown in Figure 3, Q is a large integer. For example, when the input signal is 0.813 and multiplied by the expansion factor $Q = 1000$, it becomes 813, thus realizing the error-free conversion from decimal to integer.

The complete processing of real numbers scheme realization is available at data61.csiro.au.

3.2. Convolutional Neural Network Optimization. For Mnist data, we know that the size of every picture is 28 ∗ 1 ∗ 28. "28 ∗ 1 ∗ 28" means that the image size is "28 ∗ 28" and the image has only one color channel assuming that the first layer of hidden layer nodes is 500, so a full link layer neural network will have 28 ∗ 28 ∗ 500 + 500 = 392500 parameters; when the image is bigger, the parameter will be more; this will not only lead to slow to calculate can also lead to a fitting; this time we need to solve this problem with CNN. As shown in Figure 4, a classic convolutional neural network

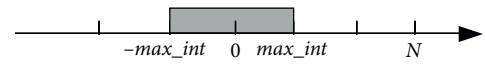

Figure 2: Schematic diagram of numerical range.

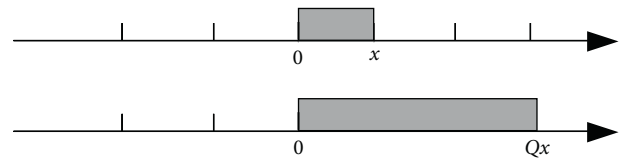

Figure 3: Numerical expansion diagram.

(CNN) architecture is shown from left to right as input layer, two convolutional and pooling layers, and full connection layers [23].

Below, we will introduce layers in CNN, respectively:

Input Layer. The input layer is the input of the entire neural network. In the convolutional neural network for image processing, it generally represents the pixel matrix of an image.

Convolutional Layer. The convolutional layer generally uses a filter to extract features with a higher degree of abstraction. The forward propagation of the filter is the process of calculating the nodes in the right identity matrix through the nodes in the small matrix on the left of Figure 5. Generally, the filter size is 3 ∗ 3 or 5 ∗ 5. Let us show you how to convert a 2 ∗ 2 ∗ 3 node matrix to a 1 ∗ 1 ∗ 5 unit node matrix. In formula (7), w represents the weight of the ith node of the output unit node matrix, and "b" represents the offset item corresponding to the ith output node. In Formula (8), "f" represents the activation function, and "g" represents the value of the ith node in the identity matrix. So, the value of the first node in the identity matrix is zero. Finally, the forward propagation of the convolutional layer structure is to move a filter from the upper left corner of the current layer to the lower right corner and calculate each corresponding identity matrix during the movement.

$$y = \sum_{x=1}^{2} \sum_{y=1}^{2} \sum_{z=1}^{3} a_{x,y,z} \times w_{x,y,z}^{i} + b^{i}, \tag{7}$$

$$g(i) = f(y). \tag{8}$$

Pooling Layer. The role of pooling layer is to effectively reduce the size of the matrix, thus speeding up the calculation speed and preventing overfitting. As shown in Figure 6, the pooling layer is mainly divided into average pooling and maximum pooling. The forward propagation of the pooling layer is also completed by moving a structure similar to the filter. However, the calculation in the pooling layer filter is not the weighted sum of nodes but a simpler operation of maximum or average value. As we hope to apply the property of homomorphic encryption to CNN, but the operation of taking the maximum value is not supported, the pooling layer of the subsequent models will all use average pooling.

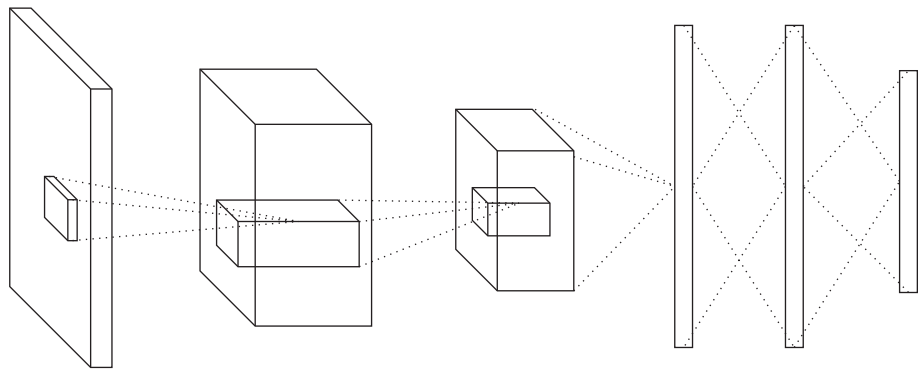

Figure 4: Convolutional neural network architecture.

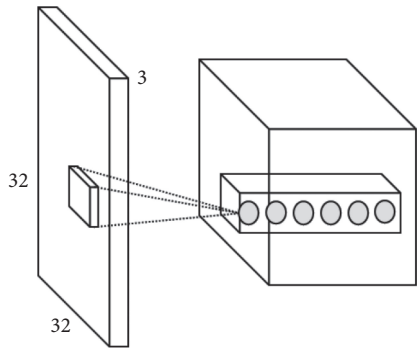

Figure 5: Convolution kernels.

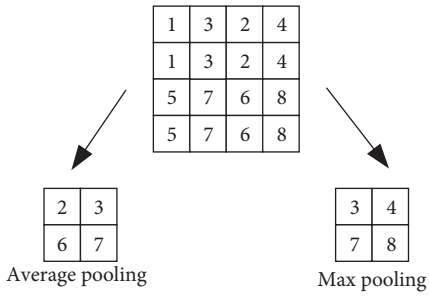

Figure 6: Average pooling and maximum pooling.

Full Connection Layer. Each neuron in this layer is connected to all neurons in the upper layer, and the input and output of each neuron are shown in Figure 7. We can regard the convolutional layer and pooling layer mentioned above as the process of automatic image feature extraction. After feature extraction, full connection layer is needed to complete classification.

Activation Layer. We can see that the convolutional layer takes the output of a unit matrix and the output of each neuron in the fully connected layer and generally inputs a weighted sum into a nonlinear function, namely, the activation function. In this way, due to the introduction of activation function, the superposition of multiple network layers is no longer a simple linear transformation, so it has a stronger performance. At first, Sigmoid function was most used (as shown in the left figure in Figure 8), but in the deep network, it had

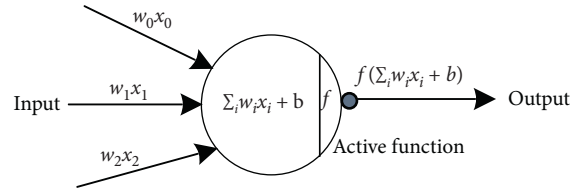

Figure 7: Schematic diagram of neuron structure.

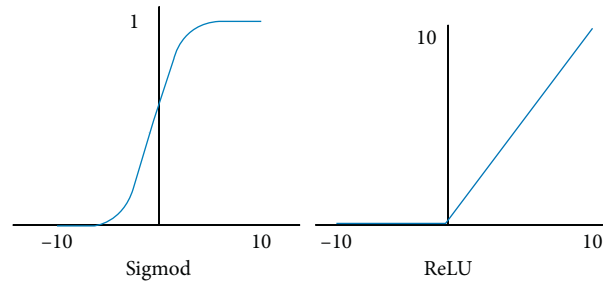

Figure 8: (a) Sigmod and (b) ReLU activation function.

gradient saturation problem. Later, scholars introduced ReLU function into AlexNet to solve the problem of gradient saturation (as shown in the right figure in Figure 6) [24, 25], and the occurrence of overfitting problem was alleviated. So, what we are going to use in our network is the ReLU activation function.

According to the above introduction of common levels in CNN, we find that the forward propagation of CNN mainly involves two kinds of operations: weighted sum and activation function. In the weighted sum, there are addition and scalar multiplication for the operations involved in the sample, which is very consistent with the property of Paillier which we introduced in the previous section. According to the description of homomorphism in the third section, we use code to encapsulate so that the ciphertext can be added directly, and scalar multiplication can be carried out, and the final decryption result is the same as the result obtained by plaintext participating in the same operation. Because the activation function is nonlinear, it does not meet the property of homomorphic encryption mentioned above. Next, we propose a solution to this problem.

4. The Practical Application

4.1. MPC. In order to combine the Paillier algorithm with CNN, we must solve the problem that the nonlinear activation function cannot satisfy the addition of the homomorphic encryption algorithm and the property of scalar multiplication homomorphism.

Thus, we draw out secure multiparty computation (which hereinafter referred to as MPC). The roots of MPC lie in a work by Yao et al. [26, 27] proposing a solution to the millionaire problem, in which two millionaires want to find out which of them is richer without revealing the amount of their wealth. In simple terms, secure multiparty computing protocol is a subdomain of cryptography, which allows multiple data owners to perform collaborative calculations without mutual trust, output the results, and guarantee that neither party can get any information other than the results it deserves. In other words, MPC technology can capture the value of data usage without revealing the original data content.

Based on the above description and considering the actual application scenario, we designed our own interactive model. As shown in Figure 9. The model owner (enterprise) can upload the trained model to the cloud; at the same time, the user also uploads the encrypted data to the cloud for inference using the model; the cloud will return the result of the ciphertext to the user; the user has a key that can be used locally for encryption and decryption.

4.2. Activation Function Processing. We define $E(x)$ and $D(x)$ as encryption and decryption, respectively, and define $A(x)$ as follows:

$$A(x) = \begin{cases} 1, & D(x) > 0, \\ 0, & D(x) \leq 0. \end{cases} \tag{9}$$

As shown in Figure 10, in the cloud model, we can remove the original layer activation function, and when the cloud model runs to the layer containing activation functions (convolution and full connection layer), the calculation result of the original convolutional layer in formula (7) or the weighted sum of the full connection layer will be returned to the local input $A(x)$ for settlement, and the calculation result of $x * A(x)$ will be returned to the next layer of cloud input network for calculation. Repeat this operation until the last layer gets the result and decrypts it locally.

4.3. Experiment. In this section, we introduce our experiment. We implement our scheme using Python3.6 on a server with Intel(R) i7 CPU, 32G RAM, Nvidia GeForce GTX 2080Ti GPU.

4.3.1. Model Training. First of all, in the CNN architecture based on Figure 11, we trained it with Mnist handwritten character dataset [27]. The Mnist dataset consists of 60,000 images of hand written digits. Each image is a 28×28 pixel array, where value of each pixel is a positive integer in the range $[0:255]$. We used the training part of this dataset, consisting of 50,000 images, to train the CNNs and the remaining 10,000 images for testing.

In Table 1, we present the main parameters in model training process. In the training process, batchsize = 128, learning rate = 0.001, convolution kernel size (3, 3, 20) and (3, 3, 50) were selected, the pooling layer filter was (2, 2), and the stride length was 1. In order to enhance the generalization ability of the model, each pixel point is divided by 255 so that the gray value of the sample is between [0, 1].

After 500 rounds of iteration, the test set can reach a high accuracy of 99.62%, as shown in Figure 12. This is of great importance to the subsequent reasoning process on the encrypted sample. Finally, we saved the trained model locally to simulate the role of the cloud service provider.

4.3.2. Model Inference Processes. In the sample of Minist datasets, each pixel on the sample is processed with the improved Paillier homomorphic encryption algorithm, and the subsequent input model is used for reasoning. However, in order to facilitate analysis, here we need to ensure that the encrypted numerical is between $[0:255]$ and only can be used to display the cipher image, and the Paillier algorithm generated in the ciphertext is generally very large, so we can perform modulo 256 operation on ciphertext data here to ensure that the gray value of each pixel of the image used for display is also between $[0:255]$. As shown in Figure 13, the first column is our original handwritten character sample, and the second column is the ciphertext sample generated after encrypting each pixel point. In fact, the encrypted image has lost its original information intuitively. However, to prevent the attacker from analyzing the original image through histogram, we further listed its histogram, as shown in the third column of Figure 13. It can be seen that the attacker cannot obtain the statistical features of the original image from the histogram. Thus, the security of uploading samples to the cloud can be guaranteed without the disclosure of private keys.

After uploading data from the user to the cloud, the output of each layer of the model interacts with the local area, and the time for encryption and decryption has the greatest impact on the time efficiency. It is necessary to note because of the cloud to maintain a persistent connection with the local, so the attacker disguised as a user sends a false data to the cloud easily to get the result of the model parameters; so, from the reasoning process, we can show the final full connection layer with a certain probability to join false neurons or random disturb neurons [27].

It can be seen that the main factor determining the reasoning time of the whole model is the amount of data to be encrypted and decrypted as well as the number of times. According to our algorithm, the amount of data here refers to the final data obtained by each layer of the model, rather than the convolution kernel weight and bias term parameters of the model itself in the reasoning process. As shown in Figure 14, we list the data size of each layer of the model after removing the activation layer from the cloud. The

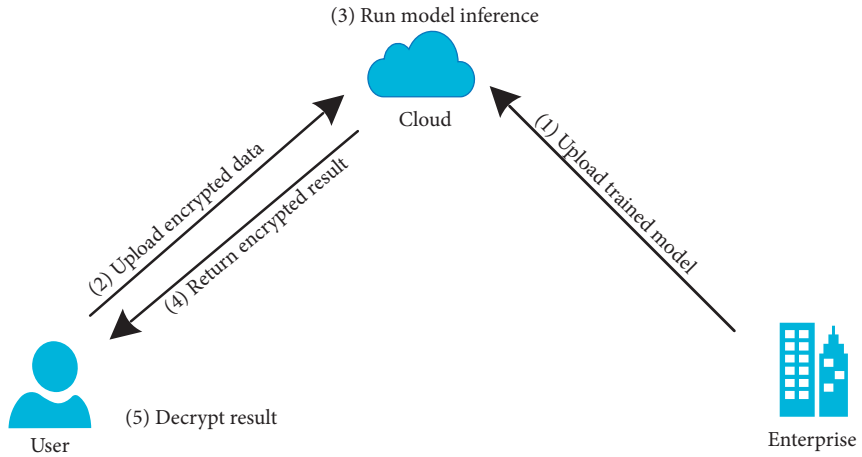

Figure 9: Data transfer architecture.

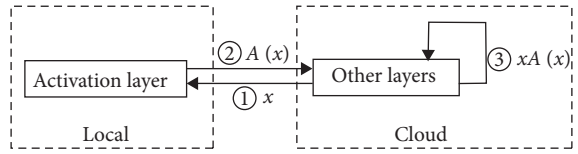

Figure 10: Data transfer architecture.

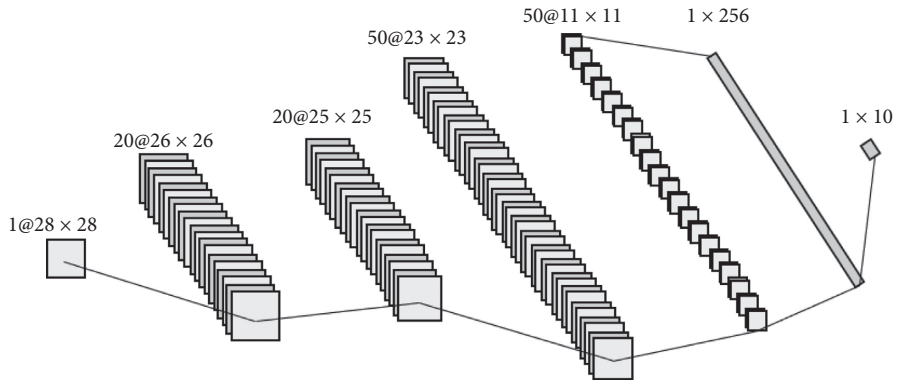

Figure 11: CNN architecture.

Table 1: Main parameters in model training stage.

Parameters	Value
Batchsize	128
Learning rate	0.001
Convolution kernel 1	(3, 3, 20)
Convolution kernel 2	(3, 3, 50)
Pooling layer filter	(2, 2)

number of encryption and decryption is related to the number of layers of the model as a whole. To be specific, we only need to encrypt the data once when it is uploaded to the cloud. The rest of the models need to interact with the local interface for decryption except for the activation layer.

The number of decrypted is equal to the number of model layers.

Next, we discuss the optimization of encryption and decryption efficiency. In order to improve the encryption and decryption time, we use GMPY2 library to optimize the homomorphism algorithm code. GMPY2 is a C-coded Python extension module that supports multiple-precision arithmetic. GMPY2 is the successor to the original GMPY module. The GMPY module only supported the GMP multiple-precision library. GMPY2 adds support for the MPFR (correctly rounded real floating-point arithmetic) and MPC (correctly rounded complex floating-point arithmetic) libraries. GMPY2 also updates the API and naming conventions to be more consistent and support the additional functionality.

We compared the time required to encrypt a sample with or without GMPY2. Table 2 shows the comparison results of homomorphic encryption efficiency when key length = 3072. It can be seen that in the case of not affecting the CPU and RAM occupancy, encryption and decryption time has a very

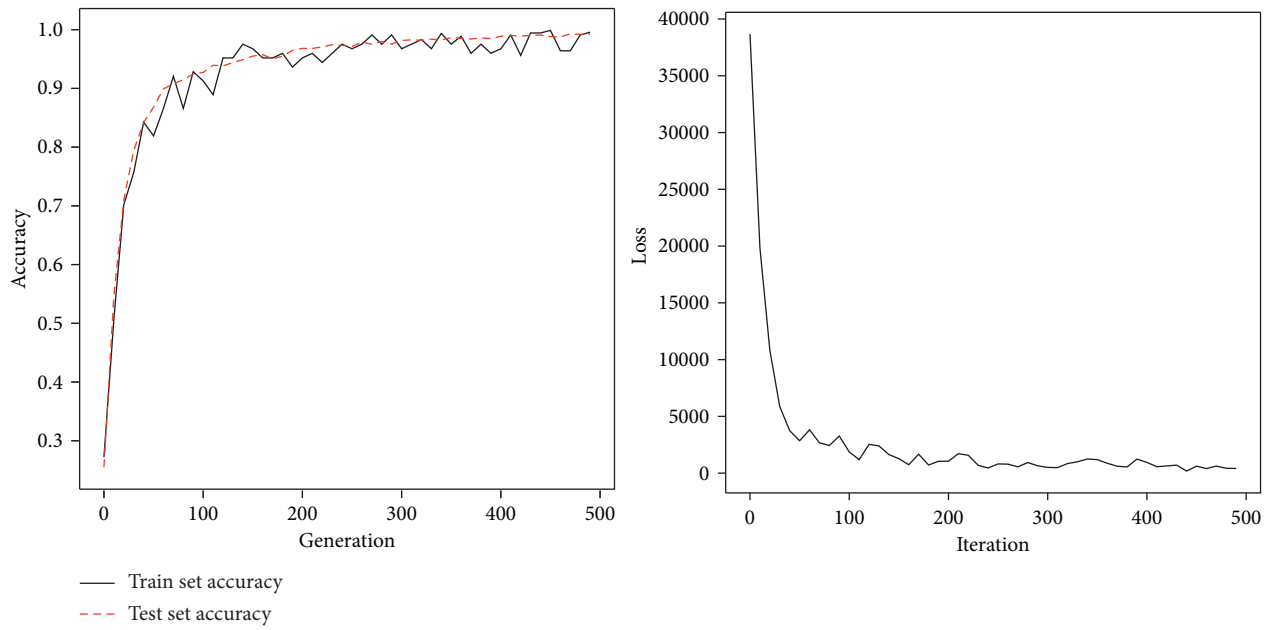

Figure 12: Accuracy and loss curve during iteration: (a) train and test accuracy; (b) loss per iteration.

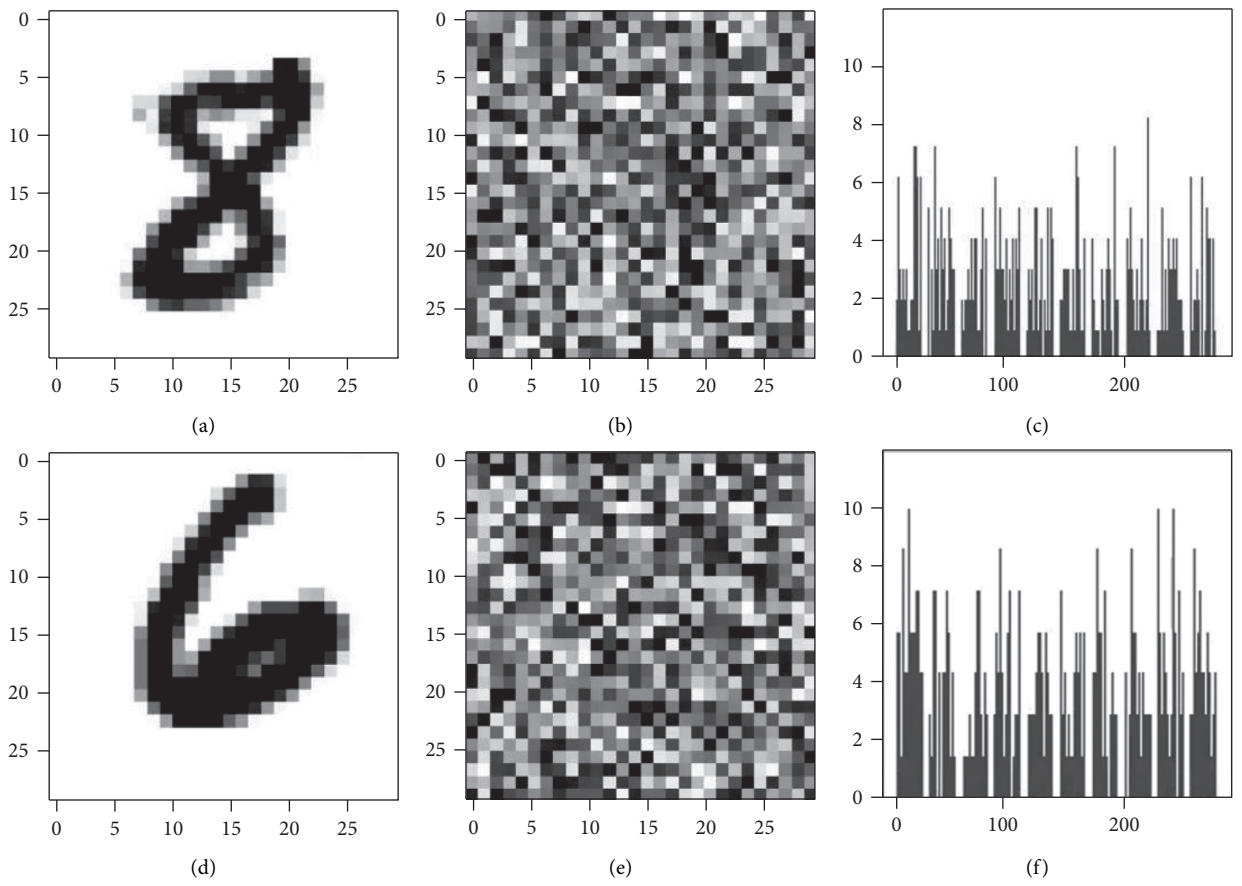

Figure 13: Sample, encrypted sample, and encrypted sample histogram.

efficient improvement. Therefore, in the following experiments, we are based on GMPY2 library optimized code for encryption and decryption work.

In addition, the selection of key length also has a great influence on the algorithm execution time. For example, Table 3 shows the time required for encryption and

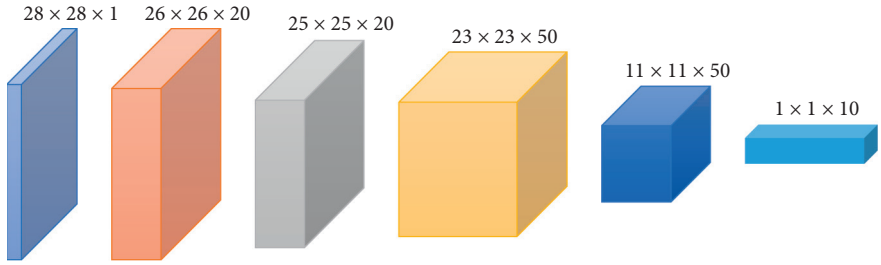

Figure 14: Different layers of data dimensions.

Table 2: When key length = 3072, the effect of GMPY2 library on the encryption efficiency of a sample.

Lib	CPU (%)	RAM (M)	Time (s)
No GMPY2	10.61	166.3	233.365
GMPY2	10.61	166.4	24.271

Table 3: The influence of key length on the efficiency of sample encryption and decryption.

Key length	512	768	1024	2048	3072
1 sample	0.281	0.589	1.163	7.707	24.399
64 sample(s)	13.175	35.506	72.886	462.563	1510.263

decryption of 1 sample and 64 samples under different key lengths. In Figure 15, we can see that the encryption and decryption time increases exponentially with the increase in key length.

We know that the current mainstream asymmetric encryption algorithm is mainly based on the difficulty of factorization of large prime numbers, and Paillier is no exception. Therefore, although the encryption and decryption time required increases rapidly with the increase in key length, the corresponding security coefficient also increases. In addition, the length of the key length in the Parillier also determines the size of the data to be processed. According to the above, the gray value of the sample will be processed to be between [0,1] before input to the model. Therefore, in the whole reasoning process of the model, the value size generated will not be too large, and for the consideration of safety, the following experiment sets the Key Length to 512. In the actual industrial production environment, the length of key length can be appropriately increased according to the requirements of security.

In Table 4, we present the model of different layers in the runtime data dimension, quantity, and the disclosure of the time needed. It can be seen that the decryption time is mainly positively correlated with the output dimension of each layer of the model, and it only takes 11.143 seconds for a sample to get the result from the cloud. Here it needs to be emphasized again that according to our model, when the trained model is uploaded to the cloud to provide service, the activation layer can be ignored. At present, the method we use to encrypt and decrypt samples is still the mode of obtaining the results through pixel-by-pixel computing. In the later stage, if parallel optimization and distributed computing are carried out on the algorithm, such as CUDA parallel programming, the operation will be transferred to GPU, so that the time efficiency will be greatly improved.

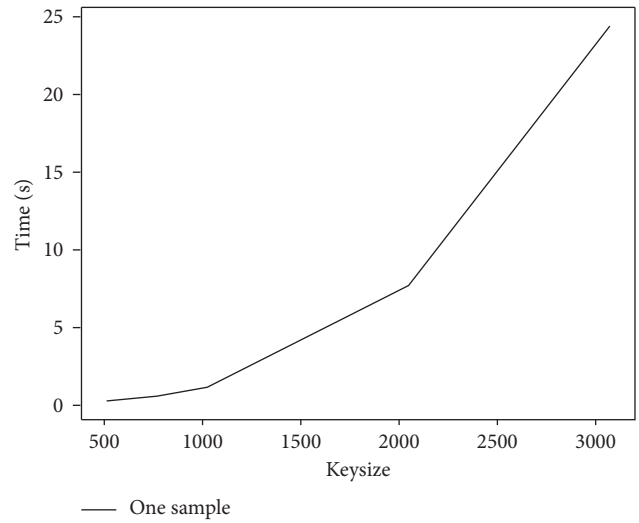

Figure 15: At different key lengths, the encryption and decryption time of one sample.

Table 4: Data dimensions, parameters, and decryption time in different layers.

Layer (type)	Output shape	Param	Decrypt time (seconds)
Input	(28, 28, 1)	0	N/A
Conv layer	(26, 26, 20)	200	2.073
Activation layer	(26, 26, 20)	0	N/A
Average pooling layer	(25, 25, 20)	0	2.511
Conv layer	(23, 23, 50)	9050	5.291
Activation layer	(23, 23, 50	0	N/A
Average pooling layer	(11, 11, 50)	0	1.211
Flatten layer	(6050)	0	N/A
Dense layer	(256)	1549056	0.055
Activation layer	(256)	0	N/A
Dense layer	(10)	2570	0.002
Activation layer	(10)	0	N/A

Total params: 1,560,876; total time: 11.143 s.

Finally, we use the ciphertext in test sets to input model for classification prediction, and the result accuracy was 99.21%. As mentioned above, the model accuracy could reach 99.62% under plaintext. In order to verify the

Table 5: Comparison of model accuracy.

Model	Our	CryptoNetsc [28]
Mnist accuray	99.21%	98.95%
Cifar-10 accuray	91.35%	Not provide
Active function	Function	Square function

Table 6: Comparison of cloud single trip prediction time.

	Our	Cryptonet	CryptoDL
Single inference time	11 s	570 s	320 s
Activation function complexity	$O(1)$	$O(n)$	$O(n)$

feasibility of the scheme on a more complex network, we also used Cifar-10 [29] dataset to train on a redesigned more complex CNN structure, and the plaintext accuracy rate reached 92.23%. Finally, ciphertext was used for reasoning, and the accuracy rate could be 91.35%. It can be seen that the scheme can adapt to more complex depth network. The decrease in accuracy in ciphertext is mainly because Paillier can handle negative number and floating-point number [30], which makes its accuracy slightly lost, but the result is still relatively good. In Table 5, after comparing with the network models which can realize ciphertext reasoning, it can be seen that almost all the models show good results on MNIST dataset. However, as the network structure becomes more complex and the computational complexity further increases, our network can still maintain a relatively good accuracy. In fact, based on our work, the accuracy itself is not highly dependent on the depth of the model network, but CryptoNets has a poor performance in the deep network.

In Table 6, we compare with the other two networks based on polynomial approximation. It can be seen that our model is far less than the other two in the total time it takes for a sample to complete the inference in the cloud. In our model, if the batchsize is increased, the required time will increase linearly. However, with the development of today's technology, the improvement of computing power and various parallelization methods, this problem can be easily solved. In addition, with the increase of network depth, the complexity of the other two models will further increase, and the time cost will also be higher.

5. Conclusions

In this paper, we combine deep learning with the homomorphic encryption algorithm and design a deep learning network model based on secure multiparty computing to ensure data privacy protection when users use the cloud deep learning model. In datasets and CNN model of varying complexity, we all got good results, which further verify the feasibility of deep learning as a service based on encrypted data. The classification accuracy rate of the encrypted two kind of dataset can reach 99.21% and 91.35%. This is a strong indication that our method can better ensure the security of users' private data in AIoT. Next, we will try to use the encrypted data to directly train CNN and then find the optimization method.

Data Availability

The data used to support the study are included within the article.

Disclosure

A preliminary study of this work was presented in the conference of "2020 International Conference on Networking and Network Applications (NaNA)."

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research work was supported by the National Joint Funds of China (U20B2050), National Key R & D Program of China (2018YFB1201500), National Natural Science Foundation of China (62072368, 61773313, and 61702411), and Key Research and Development Program of Shaanxi Province (2020GY-039, 2021ZDLGY05-09, 2017ZDXMGY-098, and 2019TD-014).

References

[1] Q. Li and Y. Zou, "A new service-oriented grid-based method for AIoT application and implementation," *Modern Physics Letters B Condensed Matter Physics Statistical Physics Applied Physics*, vol. 31, no. 19–21, Article ID 1740064, 2017.

[2] F. Tan, Y. Wang, Y. Yang et al., "A ReRAM-based computing-in-memory convolutional-macro with customized 2T2R bit-cell for AIoT chip IP applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 9, pp. 1534–1538, 2020.

[3] S. H. Seo, J. Won, and E. Bertino, "CLSC-TKEM: a pairing-free certificateless signcryption-tag key encapsulation mechanism for a privacy-preserving IoT," *Transactions on Data Privacy*, vol. 9, no. 2, pp. 101–130, 2016.

[4] G. Luo, Q. Yuan, H. Zhou et al., "Cooperative vehicular content distribution in edge computing assisted 5G-VANET," *China Communications*, vol. 15, no. 7, pp. 1–17, 2018.

[5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[6] P. S. K. S. Khodashenas, C. R. Ruiz, M. S. S. S. Siddiqui et al., "The role of edge computing in future 5G mobile networks: concept and challenges," *Cloud and Fog Computing in 5G Mobile Networks: Emerging Advances and Applications*, The Institution of Engineering and Technology, Wales, UK, 2017.

[7] A. A. Khan, M. Abolhasan, and W. Ni, "5G next generation VANETs using SDN and fog computing framework," in *Proceedings of the 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, Las Vegas, NV, USA, January 2018.

[8] O. Brdiczka, "Method and system for machine-learning based optimization and customization of document similarities calculation," *Expert Systems*, vol. 24, no. 5, 2012.

[9] J. Byrne, P. Casari, P. Eardley et al., "Reliable capacity provisioning for distributed cloud/edge/fog computing applications," in *Proceedings of the European Conference on Networks & Communications*, IEEE, Oulu, Finland, June 2017.

[10] J.-S. Chou and N.-T. Ngo, "Time series analytics using sliding window metaheuristic optimization-based machine learning system for identifying building energy consumption patterns," *Applied Energy*, vol. 177, pp. 751–770, 2016.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, vol. 1, MIT Press, Cambridge, UK.

[12] J. Gu, Z. Wang, J. Kuen et al., "Recent advances in convolutional neural networks," 2015, https://arxiv.org/abs/1512.07108v6.

[13] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the Proceedings of the ACM Conference on Computer and Communications Security (CCS) ACM*, Denver, CO, USA, Octobe 2015.

[14] R. Rivest, "On databanks and privacy homomorphism," *Foundations of Secure Computation*, 1978.

[15] J. Benaloh, *Verifiable secret-ballot elections*, Ph.D. thesis, Yale University, New Haven, CT, USA, May 1988.

[16] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[17] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the ACM Symposium on Theory of Computing(STOC 2009)*, pp. 169–178, Bethesda, MD, USA, 2009.

[18] V. Migliore, G. Bonnoron, and C. Fontaine, "Practical parameters for somewhat homomorphic encryption (SHE) schemes on binary circuits," *IEEE Transactions on Computers*, p. 1, 2018.

[19] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, *Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme*, Springer, Berlin, Germany, 2013.

[20] N. Dowlin, R. Gilad-Bachrach, K. Laine, and K. Lauter, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy," in *Proceedings of the 33rd International Conference on Machine Learning*, New York, NY, USA, 2016.

[21] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 2012.

[22] D. Stehlé and R. Steinfeld, *Making NTRU as Secure as Worst-Case Problems Over Ideal Lattices*, Springer-Verlag, Berlin, Germany, 2011.

[23] W. Zhang, "Shift-invariant pattern recognition neural network and its optical architecture," in *Proceedings of the Annual Conference of the Japan Society of Applied Physics*, Tokyo, Japan, 1988.

[24] H. Wu and X. Gu, "Max-pooling dropout for regularization of convolutional neural networks," in *Proceedings of the International Conference on Neural Information Processing*, Istanbul, Turkey, November 2015.

[25] R. Hahnloser and H. S. Seung, "Permitted and forbidden sets in symmetric threshold-linear networks," in *Proceedings of the NIPS 2001*, Vancouver, Canda, December 2001.

[26] Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164, Chicago, IL, USA, November 1982.

[27] Q. Z. Wang and L. Gao, "Neural network for processing privacy-protected data," *Journal of Cryptologic Research*, vol. 6, no. 2, pp. 258–268, 2019.

[28] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, M. Naehrig et al., "Crypto-nets: neural networks over encrypted data," *Computer Science*, https://arxiv.org/abs/1412.6181, 2014.

[29] T. Ho-Phuoc, "CIFAR10 to compare visual recognition performance between deep neural networks and humans," 2018, https://arxiv.org/abs/1811.07270v2.

[30] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: machine learning on encrypted data," in *Proceedings of the International Conference on Information Security & Cryptology*, Springer, Seoul, South Korea, November 2012.