Encrypted Data Learning and Prediction Using a BFV-based Cryptographic Convolutional Neural Network

Wei PAN*, Zepei SUN, Huanyu SANG, Zihao WANG
School of Computer Science, Northwestern Polytechnical University, Xi'an, 710072, China
panwei@nwpu.edu.cn (**Corresponding author*)

Abstract: Machine learning services are widely used for big data, cloud computing, and distributed artificial intelligence applications. Multiple parties participating in the provision of these services may access the users' sensitive data because most machine learning models use and share plaintext directly. Therefore, it is necessary to utilize cryptographic mechanisms for protecting user privacy. Homomorphic encryption provides an important information security guarantee for machine learning models. However, the complexity of fully homomorphic encryption increases with the depth of neural networks. Especially with the increase in the number of ciphertext multiplications, the time and space costs will also raise exponentially. Using homomorphic encryption in order to protect the model and data security while ensuring the computational efficiency of the employed model over encrypted data is a challenging problem. This paper proposes a BFV-based cryptographic low-latency convolutional neural network (CLOL-CNN) for solving this problem. This new network model performs deep learning and prediction over encrypted data instead of sharing plaintext data. A series of optimization operations are elaborately presented and implemented, such as cryptographic batch normalization, polynomial approximation, cryptographic convolution, and full cryptographic connection. The performance of the proposed model is evaluated with regard to its accuracy and computational overhead obtained by employing deep learning for homomorphically encrypted data. The experiments were conducted on a MNIST image dataset. The obtained results demonstrated that the proposed model has a higher accuracy and a lower time cost than other models and that it is an effective privacy-preserving deep neural network.

Keywords: Homomorphic encryption, Machine learning, Privacy-preserving network, Convolutional neural network.

1. Introduction

Machine learning service providers such as Google and Microsoft have invested a lot of energy in building machine learning models, such as Microsoft Azure and Google Prediction API. As machine learning services require more and more computer power, companies that cannot afford it will outsource machine learning tasks to the cloud. In the open cloud computing environment, transferring user data required by the machine learning model will face the risk of user privacy disclosure. Applying homomorphic encryption to the privacy protection system of the machine learning model can protect users' privacy data from being illegally stolen. Homomorphic encryption algorithms allow the machine learning model to calculate the encrypted data without decrypting the ciphertext, output the predicted ciphertext result and return it to the user, decrypted into plaintext by the user. The machine learning service providers will not obtain the user's private data, hence the security of the user's private data is guaranteed.

Homomorphic encryption algorithms are divided into partially homomorphic encryption (Rivest, Shamir & Adleman, 1978; Goldwasser & Micali, 1982) and fully homomorphic encryption algorithms (Gentry, 2009a; Fan & Vercauteren, 2012). The partially homomorphic encryption algorithms allow only a certain operation to be performed, e.g. addition or multiplication. The fully homomorphic encryption algorithms support the cryptographic execution of multiplication and addition operations, and the properties of homomorphism are satisfied for both operations. The fully homomorphic encryption algorithms mainly include Gentry's Scheme (Gentry & Halevi, 2011; Gentry, 2009b); Brakerski-Gentry-Vaikuntanathan (BGV) algorithm (Brakerski, Gentry & Vaikuntanathan, 2012) and Brakerski–Fan–Vercauteren (BFV) algorithm (Fan & Vercauteren, 2012; Brakersky, 2012), Gentry-Sahai-Waters (GSW) algorithm (Gentry, Sahai & Waters, 2013), Cheon, Kim, Kim and Song (CKKS) algorithm (Cheon et al., 2017), and so on.

The privacy-preserving computation technology represented by homomorphic encryption provides an important information security guarantee for the development and application of cloud computing, big data, and distributed artificial intelligence. The current research on privacy-preserving computation focuses on solving the security protection problem of machine learning model and on the computational efficiency of homomorphic encryption. Gilad-Bachrach et al. (2016) proposed a CryptoNets model for carrying out homomorphic encryption ciphertext calculation on the CNN model and protect users' private data. Hesamifard, Takabi & Ghasemi (2017) proposed a Cryptodl solution, which

https://doi.org/10.24846/v32i1y202304

provides a theoretical basis for finding the lowest degree polynomial approximation of nonlinear activation function within a certain error range, and reduces the time complexity of the homomorphically encrypted ciphertext and plaintext multiplication by using number theoretic transforms. Chaturvedi et al.(2018) proposed a homomorphic encryption machine learning model based on Chebyshev polynomial deep neural network. Chabanne et al. (2017) used the Taylor series to approximate the nonlinear activation function ReLu function and added a batch normalization layer to improve the machine learning model's performance. Badawi et al. (2020) proposed a homomorphic convolutional neural network (HCNN) model by using GPUs to accelerate the calculation of homomorphically encrypted ciphertext. Boemer et al. (2019) proposed a graph-HE scheme, a deep neural network model compiler developed by Intel. It supports the deployment of trained neural networks using popular machine learning frameworks to calculate homomorphically encrypted ciphertext in order to protect the user's privacy data input into the machine learning model. Hesamifard et al. (2018) proposed a privacy-preserving machine learning scheme, which required more preprocessing and calculation on the user side. Chou et al. (2018) proposed a pruning and quantization method to implement a Faster CryptoNets scheme. Due to the complexity of ciphertext operations, most schemes only carry out machine learning with small-scale network layers. If the number of homomorphic operations is too big the learning and prediction process will generate high-dimensional ciphertext data and a high level of ciphertext noises. The precision of prediction will be limited, even decryption operation may fail. Using homomorphic encryption to protect model and data security while ensuring the computational efficiency of a machine learning model over encrypted data is still a challenging problem.

Several methods were designed and implemented to improve the performance of the neural networks over encrypted data in order to improve the encrypted neural network structure and optimize the encrypted inference and computation. This paper proposes a BFV-based cryptographic low-latency convolutional neural network (CLOL-CNN) model for privacy-preserving computation. The proposed model is developed to solve the problems of low throughput prediction in the CNN model over encrypted data and high computation time of homomorphic encryption when neural networks` input data comes from different user data sources in distributed machine learning scenarios.

The model consists of BFV homomorphic encryption module, cryptographic convolution layer, cryptographic batch normalization (BN) layer, activation function layer, and cryptographic full connection layer. This model uses BFV homomorphic encryption algorithm to protect users' privacy. The data input to the neural network model is encrypted on the user side. The inner layers of the neural networks are delicately designed to support the operation of homomorphic encryption. The encoding method for plaintext is improved by scaling the plaintext weight and offset to an integer power of two and encoding a piece of plaintext data into a new monomial. The time complexity of ciphertext and plaintext multiplication is reduced. In the cryptographic convolution layer, the ciphertext is calculated and output as a homomorphic ciphertext vector through convolution operations. To improve the model's accuracy, a batch normalization layer is added to give the data input to the activation function a stable normal distribution so that the polynomial approximates the activation function in a small fixed interval. This layer does not increase the ciphertext multiplication depth. The activation function layer only performs one homomorphic encryption multiplication operation on the ciphertext vector jointly represented by all neurons. Finally, an arithmetic algorithm of homomorphic encrypted ciphertext and plaintext weight matrix is designed to reduce time cost in the cryptographic full connection layer.

This paper is organized as follows. Section 2 briefly presents the background of BFV homomorphic encryption. Section 3 provides a detailed presentation of the proposed CLOL-CNN model. Section 4 sets forth an experiment for evaluating the effectiveness of the proposed model and discusses the experimental results. Finally, Section 5 presents the conclusion of this paper.

2. Background

BFV scheme is a hierarchical fully homomorphic encryption scheme based on the Ring-Learning With Errors (RLWE) problem. The security of BFV scheme is guaranteed by very strong hardness of worst-case and average-case ideal lattice problems. BFV scheme takes polynomial rings as an algebraic system. The key, plaintext and ciphertext are polynomial forms. BFV scheme uses key exchange to limit the growth of ciphertext dimension, and uses modulus exchange to reduce noises in ciphertext. BFV's addition and multiplication homomorphic computations are more efficient than for other schemes.

2.1 BFV-based Homomorphic Encryption Scheme

The encryption steps for plaintext data based on the BFV algorithm are as follows:

Step 1: Select homomorphic encryption parameters, which include plaintext prime modulus t, ciphertext prime modulus q, polynomial highest power r, ciphertext multiplication depth L, and key exchange column number .

Step 2: Generate private key sk = s. s is a vector randomly and uniformly selected on χ, and the parameter χ is the Gaussian noise distribution in the integer field, and its value is selected to be as small as possible. The private key sk is determined by equation (1):

$$\mathrm{sk} \leftarrow \chi \tag{1}$$

Step 3: Generate public key. Randomly select a vector $a \leftarrow \left(Z_q[x]\right)/(\Phi_m(x))$, and randomly select a vector $e \leftarrow \chi$, where $Z_q[x]$ is the integer polynomial field of modulus q, x is the independent variable, and q is the coefficient module of homomorphically encrypted ciphertext polynomial, which is jointly determined by m, t, and r. $\Phi_m(x)$ is a circular polynomial, which can be decomposed into ψ(m) multiplied by the irreducible polynomials, and the public key pk is determined by equation (2):

$$\mathrm{pk} = \left(\left[-as + e\right]_q, a\right) = \left(\mathrm{pk}_0, \mathrm{pk}_1\right) \tag{2}$$

Step 4: A homomorphic encryption polynomial is used for encoding plaintext vector into a polynomial. The plaintext is encoded into a prime polynomial $M \leftarrow \left(Z_t[x]\right)/\Phi_m(x)$, where $Z_t[x]$ is the integer polynomial field of module t.

Step 5: Encrypt plaintext polynomial with the public key. Randomly generate noise polynomial $e_1 \leftarrow \chi$, and uniformly selected vector $s \leftarrow \chi$. Multiply polynomial m by an amplification factor q/t, and encrypt plaintext polynomial m with public key pk to obtain ciphertext C as it is shown in equation (3):

$$C = \left[\left[q/t\right]M + e_1 + (\mathrm{pk})s\right]_q \tag{3}$$

2.2 Operations over BFV-based Encrypted Data

Given two ciphertexts:

$$c_1 = \left[\left[q/t\right]M_1 + e_1 + (\mathrm{pk})s_1\right]_q \tag{4}$$

$$c_2 = \left[\left[q/t\right]M_2 + e_2 + (\mathrm{pk})s_2\right]_q \tag{5}$$

where M_1 and M_2 are prime polynomials encoded by two plaintexts, e_1 and e_2 are the noise polynomials randomly generated on the Gaussian noise distribution, s is the vector randomly and uniformly selected on the Gaussian distribution, and pk is the public key.

The addition result for the two ciphertexts is shown in equation (6):

$$\begin{aligned} c_1 + c_2 &= \left(\left[q/t\right]M_1 + e_1 + hs_1\right) \\ &+ \left(\left[q/t\right]M_2 + e_2 + hs_2\right) \\ &= \left[q/t\right]\left(M_1 + M_2\right) + \left(e_1 + e_2\right) + \mathrm{pk}\left(s_1 + s_2\right) \end{aligned} \tag{6}$$

The multiplication result for the two ciphertexts is shown in equation (7):

$$\left[c_1 \cdot c_2\right] = \left[q/t\right]\left(M_1 \cdot M_2\right) + e + (\mathrm{pk})^2 s_1 s_2 \tag{7}$$

When two ciphertexts are multiplied, the result is composed of three-ring elements instead of two-ring elements according to the scaling of q/t. With the increase in the number of ciphertext multiplications, the dimension of ciphertexts will become larger, and time and space costs will be increased exponentially. To avoid dimension expansion, the dimension of the ciphertext should be reduced. Let $c = [c_0, c_1, c_2]$ represent the homomorphically encrypted ciphertext calculated as $c_1 \cdot c_2$, and one should verify if $c' = [c_0', c_1']$ satisfies equation (8).

$$\left[c_0 + c_1 \cdot S + c_2 \cdot S^2\right]_q = \left[c_0' + c_1' \cdot S + R\right]_q \tag{8}$$

where R, S are small noise polynomials. When performing re-linearization, additional noise must be considered so that correct values can be decrypted.

3. Algorithmic Contributions

3.1 Overall CLOL-CNN Model

A BFV-based homomorphic encryption is employed in order to establish a cryptographic convolutional neural network model to protect the user's plaintext privacy data from being leaked. The proposed model can homomorphically encrypt users' data with different secret keys from different sources and perform deep learning prediction over encrypted data with low latency.

The framework of the CLOL-CNN model is illustrated in Figure 1. Plaintext data is encrypted by the user into a homomorphically encrypted ciphertext with a length k and input into the CLOL-CNN model. The cryptographic convolution layer outputs the ciphertext vector after convolution calculation. The cryptographic BN layer is added before the activation function layer so that the data input to the activation function has a stable normal distribution. The activation function layer applies the polynomial approximation of the nonlinear activation function to each ciphertext vector for feature combination. The cryptographic full connection layer outputs the prediction result over encrypted data.

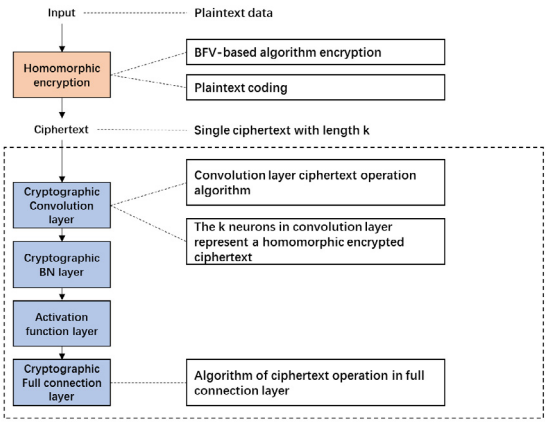


Figure 1. The framework of the CLOL-CNN model

3.2 Plaintext Encoding Method

For a given plaintext integer z, its binary expansion is expressed as $z = z^{(n-1)} \ldots z^1 z^0$. The integer z can be encoded as polynomial $f(x) = \sum_{i=0}^{n-1} b_i x^i$. If $z_i \geq 0$, then the polynomial coefficient $b_i = z_i$. If $z_i < 0$, then the polynomial coefficient $b_i = t - z_i$. The encoding method for plaintext is improved by scaling the plaintext weight and offset to an integer power of two and encoding a piece of plaintext data into a new monomial. When plaintext $b = \pm 2^k$ is selected, $k \in Z$, the binary expansion of plaintext z only has one bit with an assigned value, and the other bits are 0. The plaintext z is encoded as $b_k x^k$, which is a monomial. This plaintext encoding method gives the time complexity of the multiplication operation of homomorphically encrypted ciphertext and plaintext $O(n)$.

Based on this plaintext encoding method, the multiplication algorithm for ciphertext and plaintext is described in Algorithm 1.

Algorithm 1. Ciphertext and plaintext multiplication

1: Input: encoded plaintext $b = \mathbf{b_k x^k}$, ciphertext $c = \sum_{i=0}^{n-1} \mathbf{c_i x^i}$
2: Output: homomorphic encrypted ciphertext $\mathbf{d} = \sum_{i=0}^{n-1} \mathbf{d_i x^i}$
3: for i=0 to n-1 do
4: initialization j=i+k.
5: if $j \leq$ n-1 then
6: $\mathbf{d_j} = \left[(\mathbf{c_i b_k}) \mathbf{modulo Q} \right]$
7: else
8: $\mathbf{d_{j-n}} = \left[(\mathbf{c_i b_k}) \mathbf{modulo Q} \right]$
9: end if
10: end for
11: Calculate homomorphic encrypted ciphertext $\mathbf{d} = \sum_{i=0}^{n-1} \mathbf{d_i x^i}$

3.3 Ciphertext Operations in the Cryptographic Convolution Layer

All pixels of each image are homomorphically encrypted into a homomorphically encrypted ciphertext, which is represented by vector v. The ciphertext is decomposed into r pieces of ciphertext, represented by vectors $m^1, .., m^r$, where the i-th element of vector m^j is $v_{ó_i}(j)$. This ciphertext representation makes ciphertext convolution calculation faster and more effective in the convolution layer. When the

convolution kernel is flattened to one dimension, the convolution operation can be regarded as a restricted linear operation. When the weight vector w, convolution window size w, a set of permutations σ_i, and other parameters are set, the output of the convolution operation corresponding to the plaintext weight w_j in the convolution kernel is $\sum w_j v_{\sigma_i}(j)$. Let $m_i^j = v_{\sigma_i}(j)$ .The output of the convolution operation is calculated by $\sum w_j m^j$, i.e. it can be calculated by using r times of homomorphically encrypted ciphertext multiplication and r-1 times of homomorphically encrypted ciphertext addition. The ciphertext operations in the cryptographic convolution layer are described in Algorithm 2.

Algorithm 2. Ciphertext convolution operation

1: Input：image with size m×m, convolution kernel size n×n, step size s, filling p.

2: Output: convolution result V.

3: An image with size m×m is homomorphically encrypted into a one-dimensional vector v of length $\mathbf{m}^2$.

4: v→($\mathbf{m}^1, \ldots, \mathbf{m}^{\mathbf{n}^2}$), len($\mathbf{m}^1$)=…=len($\mathbf{m}^{\mathbf{n}^2}$) = $= \left(\left| \frac{\mathbf{m}-\mathbf{n}+2\mathbf{p}}{\mathbf{s}} \right| + 1 \right)^2$, corresponding to $\mathbf{n}^2$ weights in the convolution kernel.

5: for i=1 to $\mathbf{n}^2$ do

6: $\mathbf{m^i} = \mathbf{m^i} \times \mathbf{w_i}$

7: end for

8: for i=1 to $\mathbf{n}^2$ do

9: V=V+ $\mathbf{m^i}$

10: end for

For example, a pixel matrix $A \in R_4^4$ corresponds to an image with 4×4 pixels and a convolution filter with size 2×2. The convolution filter slides on the image with a step size of 2 in each direction. $a_{i,j}$ is expressed as row i and column j of matrix A. Firstly, the input image is homomorphically encrypted into a vector v = (a_{11}, a_{12}, a_{13}, a_{14}, a_{21}, a_{22}, a_{23}, a_{24}, a_{31}, a_{32}, a_{33}, a_{34}, a_{41}, a_{42}, a_{43}, a_{44}). The homomorphically encrypted ciphertext is decomposed into four homomorphic encrypted ciphertexts, namely $M^1 = (a_{11}, a_{13}, a_{31}, a_{33})$, $M^2 = (a_{12}, a_{14}, a_{32}, a_{34})$, $M^3 = (a_{21}, a_{23}, a_{41}, a_{43})$,$M^4 = (a_{22}, a_{24}, a_{42}, a_{44})$. The plaintext weight in the convolution kernel encodes the plaintext weight and plaintext offset into a monomial with the proposed encoding method so that the multiplication overhead of homomorphic encrypted ciphertext and plaintext is smaller than in the case of other encoding methods. The four homomorphic encrypted ciphertexts are multiplied by the corresponding four plaintext weights in the convolution kernel to calculate the four homomorphic encrypted ciphertext vectors. Then these four homomorphic encrypted ciphertext vectors are added to calculate an output vector of the convolution layer. Homomorphically encrypted ciphertext multiplication is used only four times and homomorphically encrypted ciphertext addition is used only three times in convolution operations, as it is shown in Figure 2. The homomorphically encrypted ciphertext is output as a homomorphically encrypted ciphertext vector after convolution operations.

3.4 Ciphertext Operations in the Cryptographic BN Layer

The batch normalization layer is widely used in deep learning models, and is generally placed after the convolution layer. The BN layer standardizes the distribution of all samples in a batch while keeping the statistical distribution of each sample in the batch unchanged to reduce the difference between different samples in the same batch. The BN layer is added before the activation function layer so that the data input to the activation function has a stable normal distribution. The polynomial approximation can fit the activation function in a small fixed interval to improve the model's accuracy.

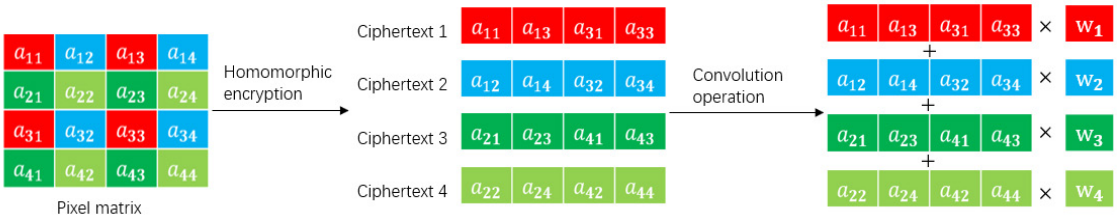


Figure 2. Ciphertext operations in the convolution layer

The ciphertext operations in the BN layer are described in Algorithm 3.

Algorithm 3. Ciphertext operation in the BN layer

1. **Input**: Homomorphic encrypted ciphertext x_i, outputted by the i-th neuron of convolution layer, its weight w, and BN layer parameters r, β
2. **Output**: batch normalized result z_{BN}.
3. Calculate ciphertext mean $\mu_z = \frac{1}{m}\sum_{i=1}^{m} w \cdot x_i$ (m is the number of neurons).
4. Calculate ciphertext variance $\sigma_z^2 = \frac{1}{m}\sum_{i=1}^{m}(w \cdot x_i - \mu_z)^2$.
5. Train the parameters $\hat{r} = \frac{r}{\sqrt{\sigma_z^2 + \varepsilon}}$, $\hat{\beta} = \beta - \frac{r - \mu_z}{\sqrt{\sigma_z^2 + \varepsilon}}$.
6. Calculate a batch normalized result $z_{BN} = (w\hat{r})x_i + \hat{\beta}$ and input it to the next layer.

The learnable parameters r, β are trained by back propagation in the training phase. The chain derivative equations of back propagation are used to calculate and update r, β and relevant weights. In forward propagation, ciphertext mean μ_z and ciphertext variance σ_z^2 are calculated by x_i, and then μ_z, σ_z^2 are updated with momentum. The new distribution value is calculated through the parameters r, β. The added BN layer may increase the computational overhead of the deep learning model. In order to apply the BN layer to homomorphic encryption, the ciphertext operations in the proposed model are calculated by the formula $z_{BN} = (w\hat{r})x_i + \hat{a}$, where $w\hat{r}$ is fixed in the prediction phase, and the multiplication depth of homomorphic encryption ciphertext is reduced to 1. The batch normalization layer is applied to improve the classification accuracy of homomorphically encrypted ciphertext, and the multiplication depth of homomorphically encrypted ciphertext is not increased.

3.5 Ciphertext Operations in the Activation Function Layer

The activation function layer only performs one homomorphic encryption multiplication operation on the ciphertext vector jointly represented by all neurons. The prediction time of the homomorphically encrypted ciphertext vector is effectively reduced. In addition, the memory bottleneck problem can be solved by homomorphically encrypting the whole image rather than the pixels at the same position of several images. This improvement enables the model to analyze larger datasets more effectively and makes the homomorphic encryption scheme more applicable in deep learning over encrypted data.

3.6 Ciphertext Operations in the Full Connection Layer

Algorithm 4. Multiplication of ciphertext vector and plaintext weight matrix in full connection layer

1. **Input**: homomorphic encrypted ciphertext vector v with length k, weight matrix W with size $k \times h$.
2. **Output**: full connection result A
3. Generate n copies of the ciphertext vector v, and define a new ciphertext vector m satisfy $m_i, m_{i+k}, m_{i+2^{n-2}\times k}, \ldots = v_i$.
4. Select the n rows of the weight matrix W to transform into a one-dimensional vector u with length $n \times k$, and then the weight matrix is encoded into h/n one-dimensional vectors $a_0, a_1, \ldots, a_{h/n}$.
5. **for** i = 1 to h/n **do**
6. $a_i = m \cdot a_i$.
7. **end for**
8. Calculate the full connection result A: rotate the h/n ciphertext vectors $a_0, a_1, \ldots, a_{h/n}$, and then add them.

Assume that the size of the full connection layer is $k \times h$. A homomorphically encrypted ciphertext vector v with length k is input into the full connection layer. A homomorphically encrypted ciphertext vector m is composed of n copies of homomorphically encrypted ciphertext vector v with length k, and the elements in homomorphic encrypted ciphertext vector m satisfy $m_i, m_{i+k}, m_{i+2^{n-2}\times k} \ldots = v_i$. A plaintext weight matrix W with n rows in the full connection layer can be transformed into a one-dimensional plaintext vector u with length $n \times k$. The homomorphically encrypted ciphertext vector m is multiplied by the one-dimensional plaintext vector u with length $n \times k$. The multiplication result is a new homomorphic encrypted ciphertext vector. One calculation can use n neurons in the full connection layer. If the full connection layer has h neurons, it only needs to perform the multiplication of homomorphic encrypted ciphertext vector and plaintext vector h/n times and output h/n vectors. A homomorphically encrypted ciphertext vector can be obtained by rotating and adding these vectors, which is the output result of the full connection layer. The calculation method is described in Algorithm 4.

4. Experiments

In order to evaluate the effectiveness of the CLOL-CNN model, the experiment carried out includes two phases: the model training phase and the homomorphically encrypted ciphertext prediction phase. The model training phase aims to train and optimize the parameters effectively used during the forward propagation of ciphertext prediction. The plaintext dataset is employed for training the model. In the ciphertext prediction phase, the plaintext data is encrypted by BFV-based homomorphic encryption, and the ciphertexts are predicted. The MNIST image dataset is applied to evaluate the proposed model. The detailed experimental environment description is shown in Table 1.

4.1 Experimental Learning and Predicting Task

The application scenario for the proposed model is shown in Figure 3. There is a learning and predicting task, which requires that the client submits the user's hand-writing pictures to the cloud and the model in the cloud performs training or prediction. User's data from different sources is encrypted into a homomorphically encrypted ciphertext. The CLOL-CNN model predicts the homomorphically encrypted ciphertext. The user decrypts the predicted result. Since the user's data is input into the CLOL-CNN model after being homomorphically encrypted, the model cannot obtain the user's private data, and the security of the user's private data is guaranteed. On the other hand, the user cannot obtain the internal structure and parameters of the CLOL-CNN model, and the security of the model is guaranteed.

4.2 Polynomial Approximation of Nonlinear Activation Function

In the proposed model, the BN layer is added before the activation function layer so that the data input to the activation function has a stable normal distribution. Because nonlinear activation functions, such as ReLU, swish, sigmoid, and tanh, are unsuitable for homomorphic encryption, the polynomial approximation method represents these functions. The polynomial approximation fit the activation function in a small fixed interval. ReLU, swish, sigmoid, and tanh functions represent the second- to fourth-order approximations of polynomials for the interval $[-4,4]$.

The polynomial approximation of the activation function on the standard normal distribution is calculated, and the polynomial regression function Polyfit in the Python software package Numpy is used in the experiment. This function input set $X=(X_1,\ldots,X_N)$,

Table 1. Experimental environment

Environment configuration	Model training	Ciphertext prediction
Processor	Amd Ryzen 7 5800H 3.20GHz	Amd Ryzen 7 5800H 3.20GHz
Operating system	Win 10 Ultimate 64 bit	Win 10 Ultimate 64 bit
Running memory	16GB	16GB
Programing language	Python	Python C++
Tool library	Tensorflow1.3, Numpy	SEAL

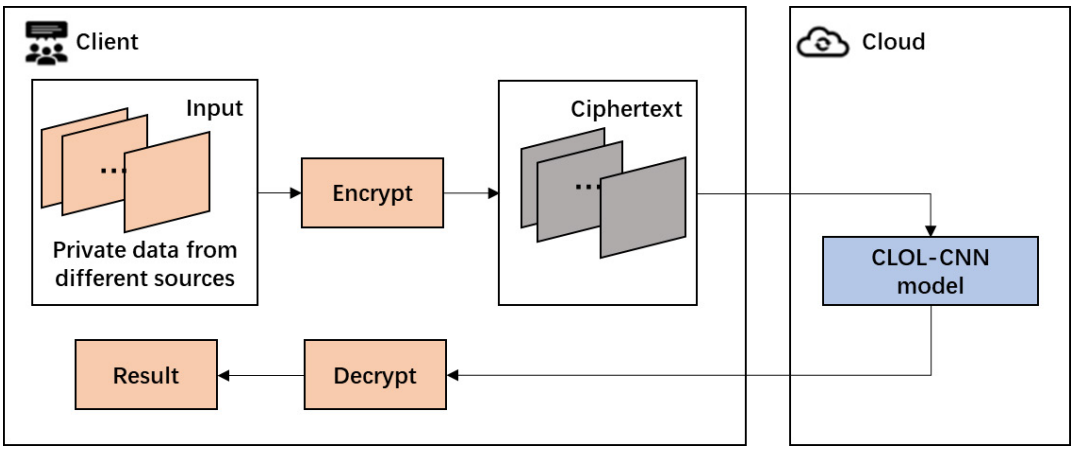
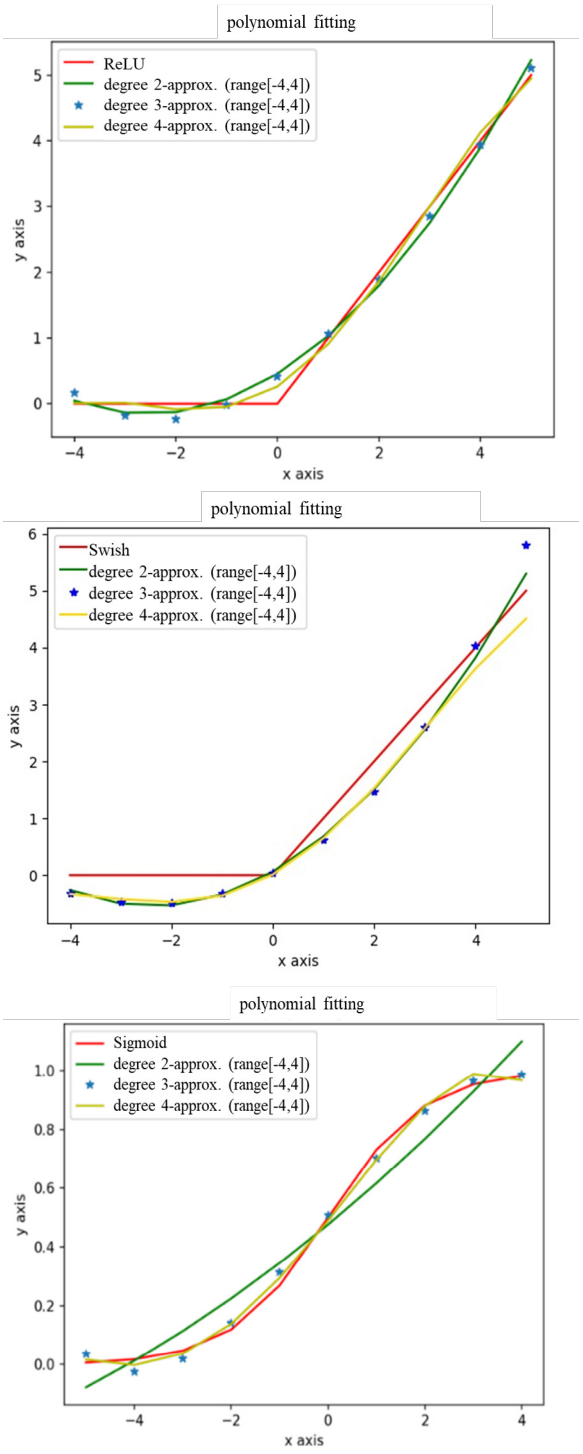


Figure 3. The application scenario for the CLOL-CNN model

and polynomial degree n, output polynomial $P(X)=C_0X+C_1X+C_2X^2+\ldots+C_nX^n$. The Polyfit function is applied to $\{(X_i,ACT(X_i))\}$, where x_i is randomly selected from the standard normal distribution with the interval [−4,4], and $ACT(X_i)$ is a nonlinear activation function. Polynomial fitting of the nonlinear activation functions is shown in Figure 4.



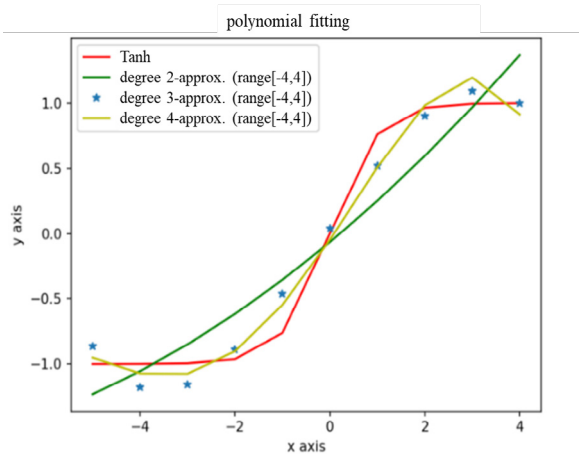


Figure 4. Polynomial function fitting of nonlinear activation functions

4.3 Training Process for MNIST Dataset

The CLOL-CNN model architecture for the MNIST dataset includes ten layers, as it is shown in Figure 5. In the training phase, the second-order, third-order, and fourth-order polynomial approximations of the four selected nonlinear activation functions are used in the activation function layer. In order to evaluate the effectiveness of polynomial approximations, 32 groups of comparative experiments were conducted. Meanwhile, the model with the BN layer and without the BN layer was tested separately to observe the influence of the BN layer on activation functions. Figure 6 displays the accuracy of the proposed model in the model training phase for the four above-mentioned training functions. When the fourth-order polynomial approximation of the ReLU function is selected as the activation function, the training accuracy of the proposed model reaches 99.56%, which is better than that of other polynomial approximations. The fourth-order polynomial approximations of the ReLU function can be further used in the ciphertext prediction phase.

4.4 Prediction Process for MNIST Dataset

The BFV-based encryption scheme in the homomorphic encryption SEAL library is used for encrypting MNIST plaintext data. In the ciphertext prediction phase, all pixels of an image are encrypted into a ciphertext vector through BFV-based homomorphic encryption.

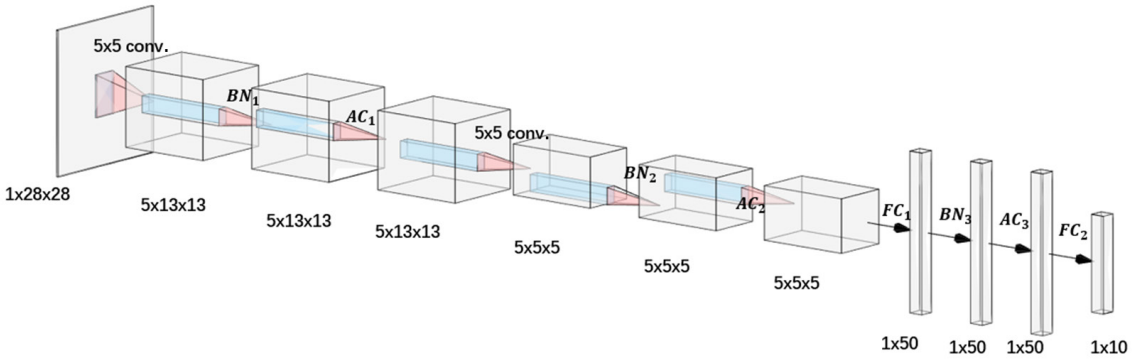


Figure 5. CLOL-CNN architecture for MNIST dataset

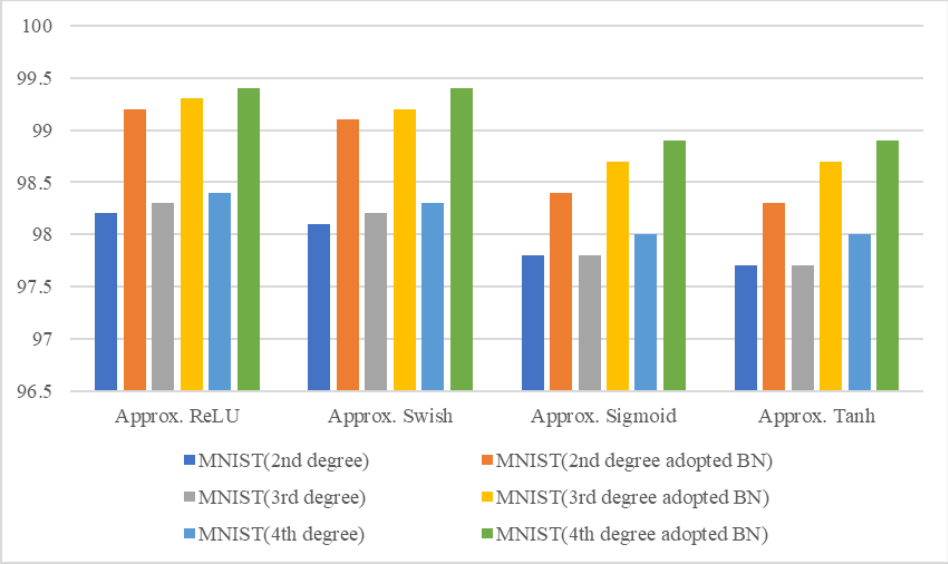


Figure 6. The accuracy of the proposed model with different polynomial approximations

Homomorphic encryption parameters are set as n = 784, t_1=1099511922689, t_2=10099512004609, $q=2^{283}-2^{33}+1$. Because $t_1 \cdot t_2 > 280$, and $q<2^{284}$, each coefficient corresponding to a homomorphically encrypted ciphertext polynomial needs 48 bytes. The setting of encryption parameters meets the requirements of homomorphic encryption security specifications and standards (Albrecht et al., 2021). Then, the prediction result over encrypted data is output by the proposed model.

4.5 Analysis of Experimental Results

To evaluate the performance of the CLOL-CNN model, nGraph-HE, CryptoNets, DeepDL, CryptoDL, and SecureML (Mohassel & Zhang, 2017) were used as baseline models and CLOL-CNN was compared with these models. The experimental results are shown in Table 2.

4.5.1 Prediction Accuracy of CLOL-CNN Model

Table 2 shows that the prediction accuracy of the CLOL-CNN model for the homomorphically encrypted MNIST dataset reaches 99.56%, which is better than other models. Compared with the other models, the CLOL-CNN model uses the BFV-based encryption scheme that does not increase the noises of the ciphertext data rapidly. In the proposed model a BN layer is added before the activation function layer so

Table 2. Comparison between CLOL model and other models

Dataset	Performance	CLOL-CNN	nGraph-HE	CryptoNets	DeepDL	CryptoDL	SecureML
MNIST	Accuracy(%)	99.56	96.9	98.95	99.3	99.52	93.4
	Runtime(s)	2.85	135	570	147	320	17.8

that the data input into the activation function can have a normal distribution and the full connection layer is retained. The CLOL-CNN model only needs to approximate the nonlinear activation function with a polynomial function in a fixed interval. The efficiency optimization algorithms can support more network layers with homomorphic operations. They are helpful in order for the model to learn more fine-grained features of encrypted data and obtain a higher classification accuracy.

4.5.2 Time Cost Required in the Prediction Phase of the Proposed Model

The proposed convolution algorithm of homomorphically encrypted ciphertext data and the full connection operation algorithm of homomorphic encrypted ciphertext vector and plaintext weight matrix optimize the calculation time for the CLOL-CNN model over encrypted data. The activation function layer only performs one homomorphic encryption multiplication operation on the ciphertext vector jointly represented by all neurons. An arithmetic algorithm of homomorphic encrypted ciphertext and plaintext weight matrix is designed to reduce time cost in the cryptographic full connection layer. In the CryptoNets model, each neuron in CNN alone represents a homomorphically encrypted ciphertext. There is an input of 845 neurons input into the CryptoNets activation function layer, representing 845 homomorphic encrypted ciphertexts. A total of 845 homomorphic encrypted ciphertext multiplications are required. In contrast to other models, the neurons of each layer in the CLOL-CNN model are collectively mapped to a homomorphically encrypted ciphertext, so the activation function layer only needs to calculate multiplication for a homomorphically encrypted ciphertext. The prediction time for a single homomorphically encrypted MNIST image is effectively reduced to 2.85s, which is lower than the prediction time for the other selected models. For the CLOL-CNN model, the runtime required for ciphertext calculation for each layer is shown in Table 3.

4.5.3 The Size of Ciphertext

In this experiment, the homomorphically encrypted ciphertext is expressed as four polynomials. The highest degree of each polynomial is 784, and each coefficient in the polynomial needs 48 bytes. All pixels of each image are encrypted into a ciphertext with the size of 28×28×48×5 bytes. A homomorphic encrypted ciphertext image requires 183.75MB of memory space. By encrypting all pixels of each image into a homomorphically encrypted ciphertext and mapping each layer of the CLOL-CNN model's neurons to a single homomorphically encrypted ciphertext vector, the memory overhead is well handled.

Table 3. Runtime required for ciphertext calculation for each layer of the CLOL-CNN model

CLOL-CNN model network layer	Time(s)
Convolution layer c_1 + BN_1	0.49
Activation function layer AC_1	0.16
Convolution layer c_2 + BN_2	0.36
Activation function layer AC_2	0.12
Full connection layer FC_1 + BN_3	1.04
Activation function layer AC_3	0.11
Full connection layer FC_2	0.57

5. Conclusion

This paper analyzes the influence of encrypted data on building deep learning models in distributed privacy-preserving applications and services. To improve the accuracy and computational overhead over homomorphically encrypted data, the BFV-based cryptographic low-latency convolutional neural network (CLOL-CNN) model is proposed, which takes the user's privacy data protection and model inference and prediction into consideration simultaneously. The structure of the model and calculation operations over encrypted data are designed in detail. The proposed model was evaluated by using the MNIST image dataset. The experimental results show that in comparison with other related models based on homomorphic encryption, the CLOL-CNN model can achieve a validation accuracy of over 99.5% with the homomorphically encrypted ciphertext of the MNIST dataset, it can accomplish the ciphertext prediction with low latency, and can improve the memory overhead of the homomorphically encrypted ciphertext.

The proposed CLOL-CNN model can effectively perform deep learning and computation on homomorphically encrypted data to protect user data privacy. As a possible future direction, the BFV-based scheme and efficiency optimization method presented in this paper could be applied in order to construct other deep learning models with more network layers. BFV-based homomorphic encryption scheme guarantees the user's privacy, data security, and model security. Cryptographic batch normalization enables the encrypted vectors to have a stable normal distribution. Polynomial approximation solves the problem related to the fact that the nonlinear activation function cannot perform homomorphic encryption. The proposed convolution and full connection operations on homomorphic encrypted data and vectors optimize the time and space cost of the deep learning model over encrypted data.

Acknowledgements

This work was supported in part through the National Natural Science Foundation of China under Grant no. U22B2025, the Key Research and Development Program of Shaanxi under Grant no. 2023-GHZD-42, and in part through the National Defense Science and Technology Key Laboratory Fund under Grant no. 61421030302012109.

REFERENCES

Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi S.,, Hoffstein, J., Laine, K., Lauter, K., Lokam, S., Micciancio, D., Moody, D., Morrison, T., Sahai, A. & Vaikuntanathan, V. (2021) *Homomorphic encryption standard*. Protecting Privacy through Homomorphic Encryption. Cham, Springer. 31-62.

Badawi, A., Chao, J., Lin, J. & Mun, C. F., Sim, J. J., Tan, B. H. M., Nan, X., Aung, K. M. M. & Chandrasekhar, V. R. (2020) Towards the AlexNet Moment for Homomorphic Encryption: HCNN, the First Homomorphic CNN on Encrypted Data with GPUs. *IEEE Transactions on Emerging Topics in Computing*, 9(3), 1330-1343. doi: 10.1109/TETC.2020.3014636.

Boemer, F., Lao, Y., Cammarota, R. & Wierzynski, C. (2019) nGraph-HE: A Graph Compiler for Deep Learning on Homomorphically Encrypted Data. In: *Proceedings of the 16th ACM International Conference on Computing Frontiers*, *CF '19*, *30 April 2019- 2 May 2019, Alghero, Italy*. New York, NY, Association for Computing Machinery. pp. 3-13.

Brakerski, Z. (2012) Fully homomorphic encryption without modulus switching from classical GapSVP. In: *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology, CRYPTO 2012, August 19-23, 2012*. Berlin, Heidelberg, Springer-Verlag. pp. 868-886.

Brakerski, Z., Gentry, C. & Vaikuntanathan, V. (2012) (Leveled) fully homomorphic encryption without bootstrapping. In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, January 8 - 10, 2012, Cambridge Massachusetts, USA*. New York, NY, Association for Computing Machinery. pp. 309-325.

Chabanne, H., de Wargny, A., Milgram, J., Morel, C. & Prouff, E. (2017) *Privacy-preserving classification on deep neural network*. IACR Cryptology ePrint Archive. https://ia.cr/2017/35 [Accessed: 6 September, 2021]

Chaturvedi, I., Ragusa, E., Gastaldo, P., Zunino, R. & Cambria, E. (2018) Bayesian network based extreme learning machine for subjectivity detection. *Journal of the Franklin Institute*, 355(4), 1780-1797. doi: 10.1016/j.jfranklin.2017.06.007.

Cheon, J. H., Kim, A., Kim, M. & Song, Y. (2017) Homomorphic encryption for arithmetic of approximate numbers. In: *Proceedings of the 23rd International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 2017*. pp. 409-437.

Chou, E., Beal, J., Levy, D., Yeung, S., Haque, A. & Fei-fei, L. (2018) *Faster CryptoNets: Leveraging sparsity for real-world encrypted inference*. ArXiv: https://arxiv.org/abs/1811.09953, 1-15. [Accessed: 8 October, 2022]

Fan, J. & Vercauteren, F. (2012) *Somewhat practical fully homomorphic encryption*. IACR Cryptology ePrint Archive. https://ia.cr/2012/144. [Accessed: 8 October, 2022]

Gentry, C. (2009a) *A fully homomorphic encryption scheme*. Ph.D. dissertation, Stanford University.

Gentry, C. (2009b) Fully Homomorphic Encryption Using Ideal Lattices. *Proceedings of the 41st ACM Symposium on Theory of Computing*, *STOC `09*, *31 May 2009- 2 June 2009, Bethesda, MD, USA*. New York, NY, Association for Computing Machinery. pp.169-178.

Gentry, C. & Halevi, S. (2011) Implementing Gentry's Fully-Homomorphic Encryption Scheme. In: *Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2011, May 15-19, 2011, Tallinn, Estonia*. pp. 129-148.

Gentry, C., Sahai, A. & Waters, B. (2013) Homomorphic encryption from learning with errors:

Conceptually-simpler, asymptotically-faster, attribute-based. In: *Proceedings of the 33rd Annual Cryptology Conference on Advances in Cryptology, CRYPTO 2013, Santa Barbara, CA, USA, 18-22 August*, *2013.* Berlin, Heidelberg, Springer-Verlag. pp. 75-92.

Gilad-Bachrach, R., Dowlin, N., Laine, K. & Lauter, K., Naehrig, M. & Wernsing, J. (2016) CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, 20-24 June 2016, New York, USA*. pp. 201-210.

Goldwasser, S. & Micali, S. (1982) Probabilistic encryption & how to play mental poker keeping secret all partial information. In: *Proceedings of the fourteenth annual ACM symposium on Theory of computing, STOC '82, 5-7 May, 1982, San Francisco, California, USA*. New York, NY, ACM Press. pp. 365-377.

Hesamifard, E., Takabi, H. & Ghasemi, M. (2017) *CryptoDL: Deep neural networks over encrypted data*. ArXiv:https://arxiv.org/abs/1711.05189. [Accessed: 8 October, 2022]

Hesamifard, E., Takabi, H., Ghasemi, M. & Wright, R. N. (2018) Privacy-preserving machine learning as a service. In: *Proceedings on Privacy Enhancing Technologies*, (2018)3, 123-142. doi: 10.1515/ popets-2018-0024.

Mohassel, P. & Zhang, Y. (2017) SecureML: A System for Scalable Privacy-Preserving Machine Learning. In: *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, *22-24 May, 2017, San Jose, CA, USA*. Los Alamitos, California, IEEE Computer Society. pp. 19-38.

Rivest, R. L., Shamir, A. & Adleman, L. (1978) A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*. (21)2, 120-126. doi: 10.1145/ 359340.359342.