A CKKS-based Privacy Preserving Extreme Learning Machine

Kunhong Li and Ruwei Huang
(Corresponding author: Ruwei Huang)

School of Computer and Electronic Information, Guangxi University
Nanning 530004, China
Email: ruweih@126.com

(Received Feb. 5, 2021; Revised and Accepted Aug. 6, 2021; First Online Oct. 22, 2021)

Abstract

As an algorithm of machine learning, extreme learning machine has good classification and prediction performance in shallow models. Still, there is a security problem of user data leakage in the cloud environment. We propose a new privacy-preserving extreme learning machine based on the CKKS homomorphic encryption (HE) scheme. As a result, users can send encrypted data to service providers for analysis and prediction without revealing their data privacy. We also designed a Single Instruction Multiple Data (SIMD) method adapted to the extreme learning machine, which reduces the number of homomorphic operations and user's waiting time. Through experiments, this paper verifies the superiority of CKKS-ELM in efficiency and accuracy.

Keywords: CKKS; Cloud Service; Extreme Learning Machine; SIMD

1 Introduction

With the development of artificial intelligence in recent years, Machine Learning as a Service (MLaaS) has been proposed, which provides Machine Learning tools as a part of cloud computing services for users to use, including image processing, speech recognition, and data analysis. However, the service provider is not necessarily honest, and it may steal the user's information. For example, the hospital needs to send the images to the cloud server for predictive analysis of some patients' symptoms. However, the data is very sensitive, and sending the images to a third party may have privacy disclosure problems.

Extreme learning machine(ELM) is a simple and efficient single hidden layer feedforward neural network learning algorithm proposed by Huang *et al.* [13], Which can quickly learn the characteristics of data sets, and has good performance in shallow classification model. It has been widely used in disease diagnosis, Traffic Sign Recognition, and image quality evaluation.

To protect data privacy, many researchers have studied the ELM for privacy protection. Samet *et al.* [22] Proposed a Privacy Protection protocol for back-propagation and ELM, based on the multi-party Secure Computing protocol they proposed, data is divided into several parts vertically or horizontally for multiple servers to process, but there are some problems such as low efficiency and large communication cost. Masato *et al.* [10,11] and proposed an ELM-based privacy preserving protocol for implementing aware agents, which protects the user's privacy. Ferhat *et al.* [5] proposed a privacy-preserving extreme learning machine based on Paillier homomorphic encryption called CPP-ELM, the training data is vertically divided into several parts, each piece of data is processed, encrypted, and transmitted to the server for integration. Yoshinori *et al.* [15] proposed a PP-ELM outsourcing scheme based on Paillier homomorphic encryption for training process of regularized ELM. Lin *et al.* [16] focuses on outsourcing ELM in cloud, and presents an optimization mechanism to improve training speed of ELM. Yang [14] proposed a privacy protection extreme learning machines based on fully homomorphic encryption scheme and carried out ciphertext image comparison experiments on medical image classification data sets.

These papers [5, 10, 11, 14–16, 22] are based on Paillier homomorphic encryption and expose the training model to the user, who can always use the model in the local environment after obtaining the model. Wang *et al.* [24] Proposed HOMO-ELM based on fully homomorphic encryption BGV scheme, which adopts HE-friendly activation function and prevents the leakage of training model. However, the amount of homomorphic operations in a single prediction is too many, which leads to a long waiting time for users and can only ensure the safety of single users.

1) We propose a privacy model scenario with high confidentiality, which is more satisfied with actual commercial application scenarios;

2) We combine the ELM with the CKKS homomorphic encryption scheme that is more suitable for floating-

point calculations and propose a better homomorphic friendly activation function to improve the classification performance;

3) According to the ELM's structure, we propose a suitable SIMD method to reduce latency during prediction.

The rest of the paper is organized as follows. In Section 2, we introduce the ELM and the CKKS fully homomorphic encryption. In Section 3, a CKKS-based ELM is proposed. In Section 4, we present experimental results and performance comparison. Finally, our conclusions are presented in Section 5.

2 Preliminaries

2.1 Extreme Learning Machine

Extreme learning machine is a single hidden layer neural network. It is composed of an input layer, a hidden layer and an output layer. The neurons of the input layer and the hidden layer, and the hidden layer and the output layer are fully connected. Let n be the neurons number of input layer, corresponding to n input features. Let l be the neurons number of the hidden layer, and let m be the neurons number of the output layer, corresponding to m classification results. Let $g(x)$ be the activation function of the hidden layer neuron, for the j-th training input data $x_j = [x_{1j}, x_{2j}, \cdots, x_{nj}]^T$, the output defines as follow:

$$T = \begin{bmatrix} t_{11} & \cdots & t_{1Q} \\ \vdots & \ddots & \vdots \\ t_{m1} & \cdots & t_{mQ} \end{bmatrix}, \tag{1}$$

$$t = \begin{bmatrix} t_{1j} \\ t_{2j} \\ \vdots \\ t_{mj} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{l} \beta_{i1} * g(w_i x_j + b_i) \\ \sum_{i=1}^{l} \beta_{i2} * g(w_i x_j + b_i) \\ \vdots \\ \sum_{i=1}^{l} \beta_{im} * g(w_i x_j + b_i) \end{bmatrix} \quad (j = 1, 2, \cdots, Q)$$

Where $w_i = [w_{i1}, w_{i2}, \cdots, w_{in}]$, represents the connection weight between the i-th neuron in the input layer and the j-th neuron in the hidden layer. $b_i = [b_1, b_2, \cdots, b_l]$, represents the bias value of the hidden layer neuron. β_{jk} represents the connection weight between the j-th neuron in the hidden layer and the k-th neuron in the output layer.

The Equation (1) can also be simplified as Equation (2):

$$H\beta = T^T \tag{2}$$

Because both w and b are randomly generated weights, and the target label matrix T is also known, the hidden layer output matrix β is uniquely determined. Training a single hidden layer neural network can be transformed into solving Equations (3):

$$\beta = H^+ T^T \tag{3}$$

where H^+ is Moore–Penrose generalized inverse of matrix H.

Because the ELM mainly reduces the error rate during training, it may cause the model to overfit. To solve this problem, Huang *et al.* [12] later introduced the regularization parameters C. Finally, the weight β can be expressed as Equations (4):

$$\beta = (\frac{I}{C} + H^T H)^{(-1)} H^T T. \tag{4}$$

Where I is the identity matrix.

2.2 Homomorphic Encryption

The concept of homomorphic encryption was proposed [21] in 1978, and homomorphic operations can be performed between homomorphically encrypted ciphertexts. For example, two plaintexts a and b are homomorphically encrypted to c and d, and the encryption function $Dec(Enc(a) \odot Enc(b)) = Dec(c \odot d) = a \oplus b$ is satisfied, where Enc is the encryption operation and Dec is the decryption operation, $\odot$ and $\oplus$ correspond to the operations on the plaintext and ciphertext domains, respectively. The early homomorphic encryption cryptosystem only supports addition or multiplication. Until 2009, Gentry [9] proposed the first fully homomorphic encryption scheme based on the ideal lattice. Homomorphic encryption is widely used in cloud environment for image comparison, face recognition, disease diagnosis, and various kind of data analyst [5, 10, 11, 14, 15, 17, 18, 22, 24]. But Cao *et al.* [4] stressed that some typical schemes are not suitable for cloud computing scenarios because the lack efficiency and do not naturally support decimal operations.

In 2017, Cheon *et al.* [7] proposed an approximate homomorphic encryption scheme called CKKS17. This scheme has greatly improved the computational efficiency of the floating-point number, which makes the use of homomorphic encryption in machine learning achieve ideal efficiency. In 2018, Cheon *et al.* [6] used the residual system as an optimization method based on the CKKS17 scheme, which improved the efficiency. Their implementation showed speed-ups 17.3, 6.4, and 8.3 times for decryption, constant multiplication, and homomorphic multiplication. This scheme is still one of the most efficient all homomorphic encryption schemes. Users can encrypt sensitive information and then send it to the service provider for analysis and processing. Finally, the user decrypts the ciphertext with his private key to obtain the analysis results.

2.3 The CKKS Leveled Fully Homomorphic Encryption Scheme

CKKS scheme is an approximate homomorphic encryption scheme. This scheme can maintain a certain precision in homomorphic operation. For example, $1.234 \times 2.222 = 2.741948 \approx 2.742$. Although the precision is decreased, CKKS is faster in operation speed, and this property is very consistent with the characteristics of machine learning. As long as the relative value is correct, the prediction results are correct. Since the module length of CKKS greatly affects the computational efficiency, Cheon *et al.* later proposed the residue number system (RNS) variant of CKKS scheme.

CKKS scheme is based on learning with errors(LWE) problem. Let N be a power of 2 integer and $R = Z[X]/(X^N+1)$ be the ring of polynomials modulo X^N+1. For a fixed base q and bit precision η, select the coprime chain of moduli $\{q_0, \cdots, q_L\}$such that $q/q_l \in (1-2^{(-\eta)}, 1+2^{(-\eta)})$. The level represents the modulus of the current ciphertext. In the leveled fully homomorphic encryption, the multiplication between ciphertexts will consume a level. When the ciphertext is in level 0, the ciphertext cannot do multiplication operation again. τ denotes a variant of the canonical embedding. The following is a brief description of the main operation part of the scheme. Readers can reference specific details and noise analysis in [6].

Setup$(q, L, \eta; 1^{(\lambda)})$**:** Given a base q, maximum computation levels L, bit precision η and security parameter λ, return power-of-two integer N, secret key distribution χ_{key}, encryption key distribution χ_{enc}, error distribution χ_{err} and RNS basis D.

KSGEN (s_1, s_2)**:** Given a ciphertext$(s_1, s_2) \in R$, sample $(a^{'(0)}, \cdots, a^{'(k+L)}) \longleftarrow U(\prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{L} R_{q_j})$ and $e^{'} \longleftarrow \chi_{err}$. Return $swk = (swk^{(0)} = (b^{'(0)}, a^{'(0)}), \cdots, swk^{(k+L)} = (b^{'(k+L)}, a^{'(k+L)})) \in U(\prod_{i=0}^{k-1} R_{p_i}^2 \times \prod_{j=0}^{L} R_{q_i}^2)$, where $b^{'(i)} \longleftarrow -a^{'(i)} \cdot s_2 + e^{'} (\mathrm{mod}\ p_i), 0 \le i < k$, $b^{'(k+i)} \longleftarrow -a^{'(k+j)} \cdot s_2 + [P]_{q_j} \cdot s_1 + e^{'} (\mathrm{mod}\ q_j), 0 \le j \le L$.

*KeyGen***:** Sample $s \longleftarrow \chi_{key}$, $(a^{(0)}, \cdots, a^{(L)}) \longleftarrow U(\prod_{i=0}^{L} R_{q_j})$ and $e \longleftarrow \chi_{err}$, return private key $sk = (1, s)$, switching key $swk \longleftarrow KSGEN(s^2, s)$, public key $pk \longleftarrow (pk^{(j)} = (b^{(j)}, a^{(j)}) \in R_{q_j}^2)_{0 \le j \le L}$.

Enc(m, pk, p)**:** Given a plaintext message $m \in \mathbb{C}^{\frac{N}{2}}$ and precision ρ, public key pk. Sample $v \longleftarrow \chi_{enc}$ and $e_0, e_1 \longleftarrow \chi_{err}$. Return $ct \in \prod_{j=0}^{L} R_{q_j}^2$, where $ct^{(j)} \longleftarrow v \cdot pk^{(j)} + (\tau^{-1}(2^p \cdot m + e_0, e_1) (\mathrm{mod}\ q_j),\ 0 \le j \le L$.

Dec(ct, sk, p)**:** Given a ciphertext message $ct \in \prod_{j=0}^{l} R_{q_j}^2$ and secret key sk, return $\tau(2^{-p} \cdot < ct^{(0}, sk >) (\mathrm{mod}\ q_0)$.

Add$(ct, ct^{'})$**:** Given a ciphertext message $ct \in \prod_{j=0}^{l} R_{q_j}^2$ and $ct^{'} \in \prod_{j=0}^{l} R_{q_j}^2$, return a ciphertext message $ct_{add} \in \prod_{j=0}^{l} R_{q_j}^2$, where $ct_{add} \longleftarrow ct + ct^{'} (\mathrm{mod}\ q_j), 0 \le j \le l$.

Mult$(ct, ct^{'})$**:** Given a ciphertext message $ct \in \prod_{j=0}^{l} R_{q_j}^2$ and $ct^{'} \in \prod_{j=0}^{l} R_{q_j}^2$, after the relinearization in RNS form, finally return a ciphertext $ct_{mult}^{(j)} \in \prod_{j=0}^{l} R_{q_j}^2$.

CMult(ct, v, p)**:** Given a ciphertext message $ct \in \prod_{j=0}^{l} R_{q_j}^2$ and vector v, $ct^{'} = \tau^{(-1)}(2^p \cdot v) \in R_{q_j}^2$, where $ct_{CMult}^{(j)} \longleftarrow ct^{(j)} \cdot ct^{'(j)} (\mathrm{mod}\ \ q_j), 0 \le j \le l$.

Rescale(ct)**:** Given a level-l ciphertext $ct = (ct^{(j)} = (ct_0^{(j)}, ct_1^{(j)}))_{0 \le j \le l}$, return $ct^{'} \in \prod_{j=0}^{l-1} R_{q_j}^2$, where $ct_i^{'(j)} = q_l^{-1} \cdot (c_i^{(j)} - c_i^{(l)}) (\mathrm{mod}\ q_j), i = \{0, 1\}$ *and* $0 \le j < l$.

3 CKKS-ELM

3.1 Model Training

The PP-ELM scheme proposed a safe model training protocol. Without exposing the data sets of all parties, the data of all parties can be aggregated, and these data can be used for training on the ELM, and finally, a complete training model can be obtained. But the premise of the original scheme is to assume that the data analyst is honest and will not steal intermediate data from the outsourced server. Let f be the dimension of data, l be the number of hidden nodes, k be the number of classes, N be the amount of record. The data provider will not leak data information when the following conditions are satisfied:

$$\begin{cases} Nf > \frac{l^2+l}{2} \\ N(f+k) > \frac{l^2+l}{2} + l \cdot k \end{cases}$$

Because the data analyst has n equations with at least $n+1$ unknown values. That is to say, each data provider needs to provide a data amount N greater than $max(\frac{l^2+l}{2f}, l)$ and carry out sum operation locally and then upload it to the cloud server, making it impossible to obtain any data even if the plaintext is decrypted. For details of sum operation, please refer to the paper [15]. Notably, the activation function used during training needs to be HE-friendly.

3.2 Tanhre Polynomial Approximation

In neural networks, nonlinear functions have stronger expressive ability than linear functions, thereby improving neural networks' classification performance. Common activation functions include Sigmoid, Tanh, and ReLu. The sigmoid function is widely used in ELM, but it compresses the data in the interval $(0, 1)$. In contrast, Tanh compresses the data in $(-1, 1)$ On the interval. Relu directly removed the negative part of the data and only kept

the positive one. Accordingly, Matthew *et al.* [19]combined the advantages of Tanh and Relu and designed a new activation function TanhRe, which has the following two characteristics: (1) Replace the negative part of Relu with Tanh, thereby improving network classification performance;(2) Negative values have boundary limits, while positive partdo not. TanhRe is defined as follows:

$$f(x) = \begin{cases} Tanh(x) & x < 0 \\ x & x \geq 0 \end{cases}$$

Dian *et al.* [20] used ELM to classify active compounds in Simplified Molecular Input Line Entry System (SMILES) and compared several activation functions. TanhRe was found to be one of the best activation functions. Therefore, TanhRe is used in this scheme to compare with previous HOMO-ELM schemes. Because the scheme based on fully homomorphic encryption only supports polynomial operation, and the common activation functions are nonlinear functions, the traditional activation function cannot be directly used under homomorphic operation, and the nonlinear activation function can only be expressed by polynomial approximation. The common approximation methods include Taylor expansion, Lagrange interpolation and least squares approximation. In this paper, the least squares approximation is adopted to ensure that the error between the approximation function and the original function in the specified interval is small. Suppose the fitting polynomial $\sum_{i=0}^{n} a_i \cdot x^i$, and the highest degree of polynomial is k. For a given n points (x_i, y_i), the problem is reduced to solving the following system of Equation (5):

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i & \cdots & \sum_{i=1}^{n} x_i^k \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \cdots & \sum_{i=1}^{n} x_i^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_i^k & \sum_{i=1}^{n} x_i^{k+1} & \cdots & \sum_{i=1}^{n} x_i^{2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \vdots \\ \sum_{i=1}^{n} x_i^k y_i \end{bmatrix} \tag{5}$$

The polynomial approximation of TanhRe is obtained by the least square method. $tr_3\ (x)$ and $tr_7\ (x)$ denote the approximate polynomial of TanhRe in the $(-5, 5)$ interval, respectively. Table 1 shows the results of TanhRe approximation. Figure 1shows a comparison between TanhRe and its polynomial approximation.

3.3 Data Encoding

To pack data into the same ciphertext for parallel calculation, the data needs to be re-encoded. w denotes the weight matrix from the input layer to the hidden layer and β denotes the weight matrix from the hidden layer to the output layer. In the homomorphic encryption scheme, let $N = 2 * len * wid, C = N/2$, where len and wid are both powers of 2. Therefore, w and β are divided into (l/C) submatrices z as follows, where l denotes the weight

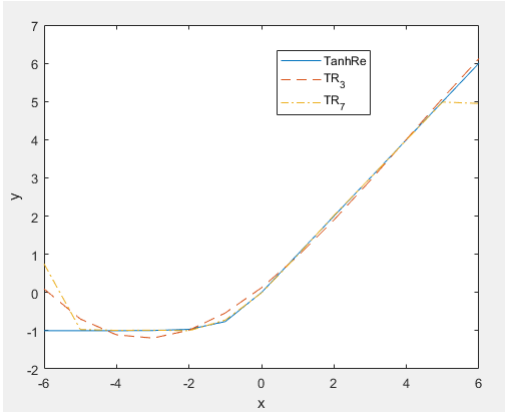

Figure 1: Comparison of Tanhre and Tanhre polynomial approximation

amount of w or β.

$$z = \begin{bmatrix} z_1 & z_2 & \cdots & z_{wid} \\ z_{wid+1} & z_{wid+2} & \cdots & z_{2*wid} \\ \vdots & \vdots & \ddots & \vdots \\ z_{(len-1)*wid+1} & z_{(len-1)*wid+2} & \cdots & z_{len*wid} \end{bmatrix}$$

Let n be the width of weight, where $n < wid$. For the 1st to $(l/C - 1)$th submatrices z_i, the $(n+1)$th to (wid)th elements of each row are filled with zero. For the (l/C) th matrix, the $(n+1)$th to (wid)th elements of the 1st to $(l\%C)$th rows are filled with zero. The$(l\%C+1)$th to (C)th rows are filled with zero. For the input data x, pack to the first n positions of the submatrix.

3.4 Horizontal SIMD

In the Homo-ELM scheme, the author encrypts the data features and network weights one by one, which requires a large number of homomorphic operations. For example, the user needs to predict an image with 64 pixels, so he needs to call the encryption method in the BGV scheme 64 times for a single image, and the obtained 64 polynomials also need to do a lot of homomorphic operations with the connection weight w. Although the Chinese remainder theorem can be used to package pixels in the same location, in practical applications, the user often does not perform too many prediction requests in the same period, so it is more practical to reduce the latency of inference stage.

3.4.1 Rotation and Summation

In CKKS scheme, by using Frobenius automorphism mapping, the ciphertext $m(x)$ corresponding to the original private key s_2 can be converted into the ciphertext $m(x^{5^i})$ corresponding to the private key s_1 by switching key swk, then the corresponding ciphertext slot space will rotate i positions to the left, and the plaintext value corresponding to the ciphertext slot changes from the original $\{x_1, x_2, \cdots, x_{N/2}\}$to

Table 1: Polynomial approximation result of TanhRe

	Result
$tr_3\ (x)$	$-0.0068661x^3 + 0.0828685x^2 + 0.7495332x + 0.1245075$
$tr_7\ (x)$	$-0.0000419x^7 + 0.0000704x^6 + 0.0022615x^5 - 0.0044224x^4$ $-0.0422012x^3 + 0.1474116x^2 + 0.8906998x - 0.0137859$

$\{x_i, x_{i+1}, \cdots, x_{N/2}, x_1, \cdots, x_{i-1}\}$, which is called rotation. For the encoded matrix vector, the method of rotation summation can be used to calculate the sum of rows or columns quickly.

Algorithm 1 Colsum

Input: A ciphertext c, an encryption of $len * wid$ matrix satisfying len is the power of 2
Output: A ciphertext $c^{'}$
$c^{'} = c$;
for $1 \leq i \leq \log_2 len$
  $cRot = Rotate(c, (-1) * 2^{i-1} * wid)$;
  $c^{'} = Add(c^{'}, cRot)$;
end for
return c

Algorithm 2 Rowsum

Input: A ciphertext c, an encryption of $len * wid$ matrix satisfying wid is the power of 2
Output: A ciphertext $c^{'}$
$c^{'} = c$;
for $1 \leq i \leq \log_2 wid$
  $cRot = Rotate(c, 2^{i-1})$;
  $c^{'} = Add(c^{'}, cRot)$;
end for
return c

3.4.2 Matrix-Vector Multiplication in Ciphertext

To obtain the prediction results for a single data, it is necessary to multiply between the weight matrix and the intermediate vector. Firstly, the column summation algorithm is needed for the data ciphertext vector x, and the x needs to be copied and filled. The matrix ciphertext is then multiplied by the data ciphertext x, and the row summation algorithm is carried out. Finally, the first column of the matrix corresponding to the ciphertext is the multiplication result of matrix and vector.

3.4.3 Multi-hidden Node Calculation

When the parameter N in the CKKS scheme is set, the capacity of a ciphertext is fixed. When ELM is applied for complex classification tasks, ELM may need a large number of hidden nodes, and multiple ciphertext matrices are required to pack all the weights for calculation.

Algorithm 3 MatrixMulVector

Input: A matrix ciphertext c, a vector ciphertext v
Output: A ciphertext $c^{'}$
$v^{'} = Colsum(v)$;
$tmp = Mult(c, v^{'})$
$tmp^{'} = Rescale(tmp)$
$c^{'} = Rowsum(tmp)$;
return $c^{'}$

Algorithm 4 Multi-hidden Node Calculation

Input: A ciphertext c, an encryption of $len * wid$ matrix satisfying len is the power of 2
Output: A ciphertext $c^{'}$
$f = \emptyset$;
for $1 \leq i \leq n$
  $c^{'} = MatrixMulVector(c_i, v)$;
  $f = Add(f, c^{'})$
end for
return f

4 Construction of Security Model

In the previous scheme based on Paillier homomorphic encryption, the CPP-ELM and the PP-ELM schemes retrieve the initial connection weights from the third-party server to the local and then use their own data set for operation and encryption. Finally, they return to the third-party server for integration, and calculate the weights from the hidden layer to the output layer. When the user predicts, the user directly obtains the trained model parameters and processes data locally. Although this protects user's data privacy and security, it exposes the entire ELM model. After the user predicts the result once, subsequent cloud services are no longer needed. For commercial service providers, this is not willing to see.

In the previous HOMO-ELM scheme, the model parameters are encrypted with a public key sp_{pk} and uploaded to the cloud server. When the user needs data prediction, the data is encrypted with sp_{pk} and uploaded to the cloud server for calculation. Finally, retrieve the calculated data locally and decrypt it with the user's private key to get the result. Although the HOMO-ELM ensures that neither the training model nor user information is disclosed, since the model in the cloud server is encrypted and uploaded by the model provider, other users who want to use it need to send their data encrypted with sp_{pk} to the server but cannot rule out malicious theft by the service provider. On the other hand, even if the

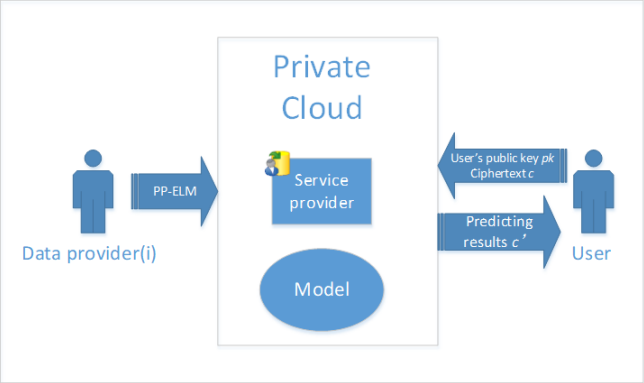

Figure 2: Operation process of CKKS-ELM

service provider does not steal user data, there is still a risk of eavesdropping when the plaintext returns to the user.

This paper proposes a privacy preserving system with better security and more suitable for commercialization. In the model training phase, we follow the PP-ELM scheme modified in Section 3.1 for model training, so that the data provider does not disclose any data, but the service provider can use the data for model training. In terms of prediction, the user generates his public and private key locally, encrypts the data with the public key, and then sends the public key and the encrypted data to the service provider. It is worth noting here that the user only needs to send the public key at the time of the first prediction, and there is no need to send the public key in later use. Then the service provider performs homomorphic operation on the encrypted data by using the ELM model and returns the encrypted data to the user, and the user decrypts the ciphertext with his private key to obtain the prediction result. Throughout the process, the data provider will not leak any data, the service provider's model will not be stolen, and neither the user's data nor the prediction results will be leaked. The process is shown in Figure 2 and the scheme comparison is shown in Table 2.

4.1 Prediction Process

4.1.1 Normalization of Data

Normalization can compress the data to the specified interval, simplify the neural network calculation and improve the classification performance. In general, the model training bases on normalized data, so the input data need to be normalized in data prediction. For the input data x, normalize it to the interval $(-1, 1)$ and do the following equation:

$$x_{Normalization} = 2 \cdot \frac{x - min(x)}{max(x) - min(x)} - 1.$$

Where $min(x)$ and $max(x)$ represent the minimum and maximum values of a feature of x, recorded by the service provider while training the model.

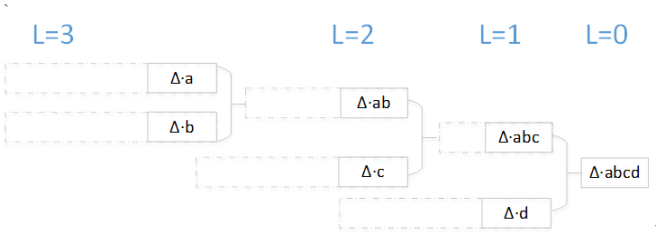

Figure 3: Multiplication in sequence

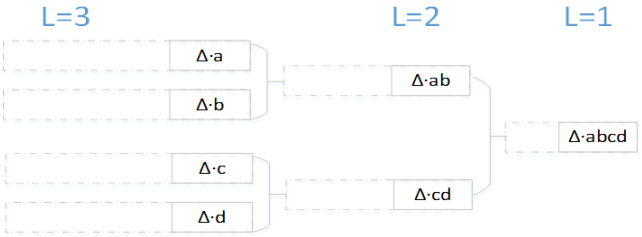

Figure 4: Multiplication in optimal sequence

4.1.2 Level Optimization

Suppose the degree of the polynomial is k. If the homomorphic multiplication operation is performed in sequence, the number of levels L that the highest degree term needs to consume is k. When the low-order term and the high-order term are added and merged, they are aligned with the lowest-order ciphertext, so the number of layers consumed by the entire polynomial operation is k. However, you can reduce the level of consumption through level optimization, and reduce the number of layers consumed from k to $\log_2 k + 1$, as shown in the Figure 3 and Figure 4:

4.1.3 Procedure

In this section, we integrate the above-mentioned to describe the complete process of data prediction phase. The concrete procedure will be given in Algorithm 5 and Algorithm 6.

5 Experiments

5.1 Experimental Setup

The experiments were carried out from the Dell computer, which has a core i5 8500 3.00 GHz CPU, 32 GB RAM, and based on Matlab 2018b and homomorphic encryption library SEAL. The main hyperparameters of SEAL are as follow: parameter $N = 16384$ of polynomial ring, levels $L = 7$, $scale = 2^{32}$, the length of q is 324bit and error standard deviation $\sigma \approx 3.2$. According to the running results of the LWE-estimator proposed in APS15 scheme [2], the ciphertext generated under the above parameter setting method meets the 128-bit security and achieve the 128-bit security against a quantum computer.

Table 2: Security comparison with three schemes

Scheme	Model protection	User data privacy protection	Multi-user safety use
PP-ELM	×	✓	✓
CPP-ELM	×	✓	✓
HOMO-ELM	✓	✓	×
Our scheme	✓	✓	✓

Algorithm 5 User Prediction

Input: A data $x \in \mathbb{R}^{feature}$
Output: Prediction classification results *predict*
//User initialize the CKKS scheme in local environment, set polynomial degree N, module Q and scale factor p, and generate public key pk, private key sk, switching key swk.
$scheme = init(params)$;
//read in data,$x = x_1, x_2, \cdots, x_{feature}$
$x = getData()$;
//encrypt x into ciphertext with pk
$ct = enc(x, pk, p)$;
if the user has not predicted or needs to change the public key **then**
  //Send ct, pk, swk and p to the service provider
  $result = ServerProcessing(ct, pk, swk)$;
else
  //Send ct to the service provider
  $result = ServerProcessing(ct)$;
end if
// Retrieve the result and decrypt the ciphertext with user's private key
$p_{res} = dec(result, sk, p)$;
// For the data with k labels, the user finds the maximum value in the first k data slots in the plaintext p_res. The corresponding index idx is the current prediction result.
$predict = MaxIn(p_{res})$;
return *predict*

Algorithm 6 ServerProcessing

Input: A ciphertext $x^{'}$,public key pk, switch key swk, and scale factor p
Output:A ciphertext $c^{'}$ with prediction result
//Gets $min(x)$ and $max(x)$ from model training and normalizes $x^{'}$
$ct = normalize(x^{'})$;
//duplicate ct vertically
$ct = colsum(ct)$;
//Get the weight w and put it into the vector according to the data encoding format, and then generate the corresponding plaintext polynomial. Finally, multiply with ct and perform a rescale operation
$IW_X = Rescale(CMult(getWeightIW(), pk, p)))$;
//duplicate IW_X horizontally
$IW_X = Rowsum(IW_X)$;
//Do an addition between IW_X and bias B
$IW_{XB} = Add(IW_X, enc(getWeightB()), pk, p))$;
//Approximate polynomial activation is performed, and H_{out} is the output of the hidden layer.
$H_{out} = poly\_activate(IW_{XB})$;
// Except for the useful information in the first column of H_{out}, the other useless information needs to be eliminated by multiplying a mask vector with H_{out}.
$H_{out} = Rescale(CMult(getMaskVec(), H_{out})))$;
//The weight LW from the hidden layer to the output layer is multiplied by H_{out} and rescaled
$LW_H = Rescale(CMult(H_{out}, getWeightLW\ ()))$;
//Finally, perform a Colsum operation to get the final result
$result = colsum(LW_H)$;
return *result*

To objectively evaluate the classification performance of the model, the data set will be randomly shuffled 10 times, and the 5-cross validation method will be used for each shuffled data set. Since the input layer's connection weights and the hidden layer in the ELM are randomly generated, which will cause the randomness of the model accuracy, each evaluation is carried out 50 times, and the average value is taken. Finally, the average value of all experimental results of the activation function is taken.

5.2 Experimental Results and Comparison

To verify the classification performance of our proposed scheme, five real-world datasets from the UCI Machine Learning Repository are used for classification prediction: Iris Data Set, Glass Identification Data Set, Ionosphere Data Set, Skin Segmentation Data Set and Landsat Satellite Data Set. The information about these datasets is shown in Table 3:

We compare the classification performance of the activation function proposed in HOMO-ELM. The first four lines of the Table 4 below are the experimental results of the model accuracy, all of which are predicted on encrypted data. The numbers in the table below represent the number of hidden nodes. $tr_3(x)$ and $tr_7(x)$ are defined in the Table 1. The activation functions proposed in HOMO-ELM with better classification performance are $x_2 = x^2 + x + 1$ and $x_3 = x^3$.

Experiments show that TanhRe activation function has better model performance than HOMO-ELM in the data

Table 3: Datasets

Datasets	Total	Features	Classes
Iris	150	4	3
Glass	214	9	6
Ionosphere	351	34	2
Satellite	6435	36	6
Skin	245057	3	2

Table 4: Classification accuracy

Method \ Datasets	$Iris_{100}$	$Ionosphere_{50}$	$Glass_{100}$	$Satellite_{100}$	$Satellite_{300}$	$Satellite_{500}$	$Skin_{100}$
x_2	96.5%	87.1%	64.34%	85.29%	86.14%	86.29%	95.86%
x_3	95.58%	83.09%	64.73%	84.24%	86.67%	87.36%	97.09%
tr_3	96.38%	88.2%	66.89%	85.49%	87.06%	88.7%	97.15%
tr_7	97.65%	87.08%	66.88%	86.43%	87.71%	89.43%	98.53%
HOMO-ELM	89.55%	-	-	78.62%	-	-	-
PP-ELM	-	-	65.4%	85%	87.5%	-	-

set with fewer features. Because CKKS scheme naturally supports the characteristic of decimal, it can approximate any nonlinear activation function for different data sets to obtain better classification accuracy, while HOMO-ELM is limited in expressing activation function because of the integer-based BGV scheme.

Let the number of hidden nodes be L, the number of data features be F, the number of classes be K and the highest degree of activation functions as T. From the perspective of efficiency, the complexity of homomorphic operations for each data prediction by HOMO-ELM is $\mathcal{O}(LF + KF + \log_2 T)$ and the number of ciphertexts for communication is $F + K$. Let the highest degree of the polynomial ring of CKKS is N, the complexity of homomorphic operations for each data prediction in our scheme is $\mathcal{O}(\frac{LF+KF}{N} + \log_2 T)$ and the number of ciphertexts for communication is $\lceil \frac{F}{N} \rceil + \lceil \frac{K}{N} \rceil$, which significantly reduced latency for user prediction. Table 5 summarizes benchmark results of our experiments.

For more comparisons, the accuracy of skin data set in PP-LG results from the classification test of Logistic model under Privacy Protection proposed by Aono *et al.* [3] is 93.89%. F. Al-Obeidat *et al.* [1] achieved 88.29% accuracy on their proposed optimized method while achieving 82% and 85.71% on ID3 and C4.5 respectively. DU *et al.* [8] proposed a privacy preserving optimized k-NN algorithm and the highest accuracy of Iris and skin segmentation dataset is 97% and 98%, which is closed to our scheme on tr_7 method. But they need a few seconds to interfere with a single input. Sarpatwar *et al.* [23] proposed a systematic approach to perform multiplicative depth constrained knowledge distillation that enables efficient encrypted inference. They carried out on two other UCI Dataset: Bank Management and MAGIC Gamma Telescope, and we achieve a closed result in accuracy but very low latency since their SIMD strategy need lots of computation consumption, which leads to approximately 5 minutes. The above facts all show that the privacy preserving extreme learning machine scheme proposed in this paper not only guarantees certain accuracy in the lightweight privacy preserving model but also has lower computation cost.

6 Conclusions

In this paper, a new privacy preserving extreme learning machine scheme called CKKS-ELM is proposed. In the aspect of homomorphic encryption, we use the CKKS scheme, which is more suitable for machine learning applications. In the aspect of ELM, we propose horizontal SIMD processing method and better activation function. In communication construction, we propose a privacy preserving system that is more suitable for commercialization. Extreme learning machine has been studied as a classifier with good classification performance. The future research direction can combine fully homomorphic encryption with the new extreme learning machine theory while taking into account efficiency and accuracy.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (No.62062009), the Science and Technology Major Project of Guangxi of China (AA17204058-17) and the Science and Technology Major Project of Guangxi of China (Guike AA18118047-7). The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

Table 5: Runtime in different dataset and setting (Seconds)

Method \ Datasets	$Iris_{100}$	$Ionosphere_{50}$	$Glass_{100}$	$Satellite_{100}$	$Satellite_{300}$	$Satellite_{500}$	$Skin_{100}$
x_2	0.126	0.16	0.143	0.164	0.391	0.505	0.126
x_3	0.151	0.2	0.18	0.211	0.525	0.685	0.15
tr_3	0.159	0.22	0.187	0.217	0.54	0.706	0.157
tr_7	0.322	0.257	0.291	0.322	0.843	1.128	0.245

References

[1] F. Al-Obeidat, A. Al-Taani, N. Belacel, L. Feltrin, and N. Banerjee, "A fuzzy decision tree for processing satellite images and landsat data," *Procedia Computer Science*, vol. 52, pp. 1192–1197, 2015.

[2] M. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.

[3] Y. Aono, T. Hayashi, L. Phong, and L. Wang, "Privacy-preserving logistic regression with distributed data sources via homomorphic encryption," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 8, pp. 2079–2089, 2016.

[4] Z. Cao, L. Liu, and Y. Li, "Ruminations on fully homomorphic encryption in client-server computing scenario," *International Journal of Electronics and Information Engineering*, vol. 8, no. 1, pp. 32–39, 2018.

[5] F. Çatak and A. F. Mustacoglu, "CPP-ELM: Cryptographically privacy-preserving extreme learning machine for cloud systems," *International Journal of Computational Intelligence Systems*, vol. 11, no. 1, pp. 33–44, 2018.

[6] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *International Conference on Selected Areas in Cryptography*, pp. 347–368. Springer, 2018.

[7] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 409–437. Springer, 2017.

[8] J. Du and F. Bian, "A privacy-preserving and efficient k-nearest neighbor query and classification scheme based on k-dimensional tree for outsourced data," *IEEE Access*, vol. 8, pp. 69333–69345, 2020.

[9] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 169–178, 2009.

[10] M. Hashimoto and Q. Zhao, "An ELM-based privacy preserving protocol for implementing aware agents," in *2017 3rd IEEE international conference on cybernetics (CYBCONF)*, pp. 1–6. IEEE, 2017.

[11] M. Hashimoto and Q. Zhao, "A privacy preserving protocol for cloud-based implementation of aware agents," *Journal of Information Processing*, vol. 26, pp. 486–496, 2018.

[12] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2011.

[13] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[14] Y. Jing. *Research on Key Technologies of Homomorphic Encryption*. PhD thesis, University of Electronic Science and Technology of China, 2019.

[15] S. Kuri, T. Hayashi, T. Omori, S. Ozawa, Y. Aono, L. Wang, S. Moriai, et al., "Privacy preserving extreme learning machine using additively homomorphic encryption," in *2017 IEEE symposium series on computational intelligence (SSCI)*, pp. 1–8. IEEE, 2017.

[16] J. Lin, J. Yin, X. Zhang, Z. Cai, and Y. Ming, "Optimization mechanism for secure outsourcing extreme learning machine in cloud computing," *Computer & Digital Engineering*, vol. 47, no. 1, pp. 157–160, 2019.

[17] L. Liu and Z. Cao, "Analysis of two confidentialitypreserving image search schemes based on additive homomorphic encryption," *International Journal of Electronics and Information Engineering*, vol. 5, no. 1, pp. 1–5, 2016.

[18] Y. Ma, L. Wu, X. Gu, J. He, and Z. Yang, "A secure face-verification scheme based on homomorphic encryption and deep neural networks," *IEEE Access*, vol. 5, pp. 16532–16538, 2017.

[19] M. Maimaitiyiming, V. Sagan, P. Sidike, and M. T. Kwasniewski, "Dual activation function-based extreme learning machine (elm) for estimating grapevine berry yield and quality," *Remote Sensing*, vol. 11, no. 7, p. 740, 2019.

[20] D. E. Ratnawati, Marjono, Widodo, and S. Anam, "Comparison of activation function on extreme learning machine (elm) performance for classifying the active compound," in *AIP Conference Proceedings*, vol. 2264, p. 140001. AIP Publishing LLC, 2020.

[21] R. L. Rivest, L. Adleman, M. L. Dertouzos, et al., "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[22] S. Samet and A. Miri, "Privacy-preserving back-propagation and extreme learning machine algo-

rithms," *Data & Knowledge Engineering*, vol. 79, pp. 40–61, 2012.

[23] K. Sarpatwar, K. Nandakumar, N. Ratha, J. Rayfield, K. Shanmugam, S. Pankanti, and R. Vaculin, "Efficient encrypted inference on ensembles of decision trees," *arXiv preprint arXiv:2103.03411*, 2021.

[24] W. Wang, Y. Gan, C. M. Vong, and C. Chen, "Homo-elm: fully homomorphic extreme learning machine," *International Journal of Machine Learning and Cybernetics*, pp. 1–10, 2020.

Biography

Kunhong Li is a master degree student in the school of Computer and Electronic Information, Guangxi University. His research interests focus on information security.

Ruwei Huang received her Ph.D from Xi'an Jiaotong University in 2012. She is currently an associate professor of the school of Computer and Electronic Information, Guangxi University. Her research interests include cloud computing and fully homomorphic cryptography.