Hindawi
Security and Communication Networks
Volume 2020, Article ID 5328059, 9 pages
https://doi.org/10.1155/2020/5328059

Research Article

SecureBP from Homomorphic Encryption

Qinju Liu,[1,2] **Xianhui Lu**,[1,2] **Fucai Luo**,[1,2] **Shuai Zhou**,[3] **Jingnan He**,[1,2] **and Kunpeng Wang**[1,2]

[1]*State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing 100093, China*
[2]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China*
[3]*Faculty of Engineering and Information Technology Engineering, University of Technology Sydney, Ultimo, NSW 2007, Australia*

Correspondence should be addressed to Qinju Liu; liuqinju@iie.ac.cn

Received 18 November 2019; Revised 16 May 2020; Accepted 26 May 2020; Published 12 June 2020

Academic Editor: Stelvio Cimato

We present a secure backpropagation neural network training model (SecureBP), which allows a neural network to be trained while retaining the confidentiality of the training data, based on the homomorphic encryption scheme. We make two contributions. The first one is to introduce a method to find a more accurate and numerically stable polynomial approximation of functions in a certain interval. The second one is to find a strategy of refreshing ciphertext during training, which keeps the order of magnitude of noise at $\widetilde{O}(e^{33})$.

1. Introduction

Driven by massive amounts of data and the high scalability, versatility, and high efficiency of cloud computing, modern machine learning (ML) has been widely used in many fields, including health care, military, and finance [1–3]. These fields often contain a large amount of sensitive data, so how to protect the data privacy while using them becomes a very important problem. At present, there exist various approaches that can be used to protect data privacy. Differential privacy (DP), secure multiparty computation (MPC), and homomorphic encryption (HE) are the most widely used methods for this problem.

DP allows one to control the amount of information leaked from an individual record in a dataset. By using DP, one can ensure privacy for any entity whose information is contained in the dataset as well as to create models that do not leak this information about the data they were trained on. Therefore, DP is mainly used in the training process. However, we are more concerned about how to use cryptographic methods to protect data privacy.

Most MPC methods establish a communication protocol among the parties involved such that if the parties follow the protocol, then they will end with the desired results while protecting the security and privacy of their respective assets [4–7]. However, due to the large scale of data used in machine learning, the communication cost of MPC is very high.

HE is also another major method to protect data privacy, which allows us to perform certain arithmetic operations on encrypted data without decryption. Fully homomorphic encryption (FHE) (it allows us to perform arbitrarily complex and efficiently computable evaluations over encrypted data without decrypting them) was originally introduced by Rivest et al. in 1978 [8]. But it had been an open problem until Gentry presented the first plausible candidate FHE construction based on ideal lattices in 2009 [9]. Since then, a series of works [10–15] have been proposed to improve the security assumptions and efficiency of FHE, following Gentry's blueprint. Currently, some public libraries are available (Table 1), namely, HElib [16] and SEAL [17] based on BGV scheme [18] and FHEW [19] and TFHE [20, 21] based on GSW scheme [14] and HEAAN [22] based on CKKS scheme [23]. The BGV-based schemes can handle a lot of bits at the same time, so they can pack and batch many operations in the SIMD manner. However, the set of operations that are efficient with BGV depends on the section of the parameter set. The GSW-based schemes can use Boolean circuits to deal with nonlinear operations quickly, but their computational efficiency of arithmetic

Table 1: Outstanding performance of HE schemes.

Based scheme	Library	Plaintext	Operation
BGV	HElib, SEAL	Finite field packing	Addition, multiplication
GSW	FHEW, TFHE	Binary string	Look-up table
CKKS	HEAAN	Real/complex packing numbers	Fixed-point arithmetic

operations is relatively low. The CKKS-based schemes can perform efficient approximate arithmetic operations on encrypted data by introducing a novel encoding technique and a fast rescale operation, but they cannot deal with nonpolynomial operations. It is widely used in machine learning due to its high efficiency in arithmetic operations (which is why we chose the CKKS scheme for our SecureBP model).

Two important use-cases for machine learning models are predictions-as-a-service (PaaS) setting and training-as-a-service (TaaS) setting. In the PaaS setting, a large organization (or the cloud) uses its proprietary data to train machine learning models. The organization now hopes to monetize the model by deploying services that allow users to upload their inputs and receive predictions for price. In the TaaS setting, the organization makes profits by deploying services that allow users to upload their encrypted inputs and receive the encrypted machine learning model. Moreover, in this setting, since the process of training an encrypted model is time- and resource-consuming, the techniques and proprietary tools for the training algorithm are often considered critical intellectual property by its owner, who is typically not willing to share them.

BP [24] is one of the most classical and widely used neural network models. It is more powerful than linear regression and logistic regression models. Moreover, the BP network already has the basic module of deep neural network (DNN); in other words, the BP network is the cornerstone of DNN. Therefore, when we study the data privacy protection of machine learning, it is appropriate to take the BP network model as the breakthrough point.

1.1. Our Contributions. In this paper, we present a secure backpropagation neural network model (SecureBP) based on HE. In this model, in a setup phase, the data owner (user) encrypts his data and sends them to the cloud. In the computation phase, the cloud can train the model on the encrypted data without learning any information beyond the ciphertext of data. Technically, we have two main contributions: a more accurate polynomial approximation technique and a lightweight interactive scheme to refresh ciphertexts during training.

We focus on the TaaS setting in this paper and we choose HE (i.e., CKKS scheme) as the method to protect user's data. For clarity, let us review the technical challenges and difficulties of using HE for the BP network in the TaaS setting. Firstly, in the BP network, each node is activated before output by an activation function, which is usually selected by nonpolynomial functions, such as sigmoid, tanget-hyperbolic (tanh), or rectified linear unit (ReLU). However, most existing HE schemes is that they only support polynomial arithmetic operations. The evaluation of the activation function is an obstacle for the homomorphic implementation of the BP network since it cannot be expressed as a polynomial. In addition, in order to ensure security, HE introduces some noise in encryption, and the noise increases as the homomorphic computation proceeds. When the noise reaches a certain threshold, the decryption error will occur. Therefore, in view of the abovementioned technical difficulties, we make the following two contributions.

The first contribution is that by using Chebyshev polynomials (in fact, several studies have suggested this approach, but none have examined it in detail), we introduce a more accurate polynomial approximation $L_n(x)$ of sigmoid function for a certain interval. Compared with Taylor polynomials, our method causes more similarities of derivatives with the sigmoid function (see Section 3.1).

The second contribution is that we propose a lightweight interaction protocol, which is a novel strategy to refresh ciphertext during training. The trivial way to deal with the growing noise is bootstrapping. However, bootstrapping comes with high computational overhead. To avoid costly bootstrapping of HE, we present the lightweight interaction protocol during training. By this method, on the one hand, no technical information of the cloud training model is provided to the user. On the other hand, the noise of weight ciphertext grows linearly after it grows to a certain value.

Now that the basic ingredients are in place, we construct our SecureBP network. To demonstrate the feasibility of our SecureBP, we estimate its performance on three datasets: Iris dataset, Diabetes dataset and Sonar dataset (see Table 2), which are from the University of California at Irvine (UCI) dataset repository [25].

1.2. Related Work. Before the current work, there have been some researches on privacy-preserving machine learning algorithm [26–29]. These papers propose solutions based on MPC and HE techniques (see Table 3), but they appear to incur some problems.

Privacy-preserving machine learning via MPC provides a promising solution by allowing different parties to train various models on their joint data without revealing any information beyond the outcome. They require interactivity between the party that holds the data and the party that performs the blind classification. Even though practical performances of MPC-based solutions have been impressive compared to FHE-based solutions, they incur other issues such as network latency and high bandwidth usage. Because of these downsides, HE-based solutions seem more scalable for real-life applications.

Privacy-preserving machine learning based on HE is more challenging. As we mentioned before, the standard

Table 2: Performance of SecureBP in time and accuracy.

Dataset	Accuracy (%)	Time (ms)
Iris	79.6	7632.7
Diabetes	65.1	9962.1
Sonar	82.23	2.0993×10^8

activation function is a challenge in applying HE to the machine learning algorithm. Faced with this challenge, Ran Gilad-Bachrach et al. [30] propose a solution (CryptoNets) where instead of standard activation function, they use a square function. Homomorphic computation depends on the total number of levels required to implement the network and results in a relatively high computational overhead which bounds CryptoNets practicability in resource-limited settings where the data owners have severe computational constraints. Moreover, the inherent limitation of most existing HE constructions is that they only support the arithmetic operations over modular spaces. Therefore, their approaches required the size of parameter for real number operations (i.e., no modular reduction over plaintext space) which is too large to be practically implemented.

1.3. Organization. Section 2 briefly introduces some notations and reviews the framework of BP. Section 3 describes our SecureBP model. In section 4, we estimate our model and discuss the estimation and implementation results.

2. Preliminaries

2.1. Notations. All logarithms are base 2 unless otherwise indicated. During homomorphic operations, we use $\otimes$ to denote the multiplication between ciphertexts; $\oplus$ denotes the addition between ciphertexts and $\odot$ denotes the scalar multiplication between a constant and a ciphertext.

Next, we introduce some signs used in the BP network:

(i) $\{x_i\}_{i\in[m]}$, the input value.
(ii) $\{h_j\}_{j\in[z]}$, the output value of hidden layer.
(iii) $\{O_k\}_{k\in[d]}$, the output value of output layer.
(iv) $\{W^h_{j,i}\}_{j\in[z],i\in[m]}$, the weight connecting the hidden-layer j-th node and the input-layer i-th node.
(v) $\{W^O_{k,j}\}_{k\in[d],j\in[z]}$, the weight connecting the output-layer k-th node and the hidden-layer j-th node.
(vi) $\{b^h_j\}_{j\in[z]}$, the bias of hidden-layer j-th node.
(vii) $\{b^O_k\}_{k\in[d]}$, the bias of output-layer k-th node.
(viii) L, the learning rate.

2.2. The Framework of BP. In this subsection, we give a brief review of one version of the BP network. For ease of presentation, in this paper, we only consider a neural network of three layers (input layer, hidden layer, output layer). It is trivial to extend our work to the multilayers network. This configuration can be seen from Figure 1.

In the BP algorithm, there is one forward phase and one backward phase during each iteration. Then, the whole BP algorithm can be described in Algorithm 1.

The forward phase starts from the input layer and approaches the output layer. During this phase, weighted sums and activations are computed for every node in each layer using the activation function, which is normally the sigmoid function. That is,

$$h_j = \sigma\left(\sum_{i=1}^{m} W^h_{j,i} \cdot x_i + b^h_j\right), \quad \text{(in hidden layer)},$$
$$O_k = \sigma\left(\sum_{j=1}^{z} W^O_{k,j} \cdot h_j + b^O_k\right), \quad \text{(in output layer)}. \tag{1}$$

The backward phase starts from the output layer and descends toward the bottom layer (i.e., the input layer) of the network to compute gradients. Finally, we need to update the weights ($\{W^h_{j,i}\}\{W^O_{k,j}\}$) and biases ($\{b^h_j\}$, $\{b^O_k\}$) using the computed gradients. The rules for updating are as follows:

$$\begin{aligned} W^O_{k,j} &= W^O_{k,j} + l \cdot \mathrm{Err}^O_k \cdot h_j, \\ b^O_k &= b^O_k + l \cdot \mathrm{Err}^O_k, \\ W^h_{j,i} &= W^h_{j,i} + l \cdot \mathrm{Err}^h_j \cdot x_i, \\ b^h_j &= b^h_j + l \cdot \mathrm{Err}^h_j, \end{aligned} \tag{2}$$

where $\mathrm{Err}^O_k = O_k(1 - O_k)(t_k - O_k)$ and $\mathrm{Err}^h_j = h_j(1 - h_j)\sum_k \mathrm{Err}^O_k W^O_{k,j}$.

3. SecureBP Based on CKKS Scheme

In this section, we explain how to securely train the BP network model using the CKKS scheme.

3.1. A Decent Polynomial Approximation. In the preceding update formula, except for activation function inside neurons (i.e., sigmoid function $\sigma(x) = (1/1 + e^{-x})$), all other operations in BP network are addition and multiplication, so they can be implemented over encrypted data. One limitation of the existing HE schemes is that they only support polynomial arithmetic operations. The evaluation of the activation function is an obstacle for the implementation of the BP network since it cannot be expressed as a polynomial. Hence, in order to operate a complete BP neural network over encrypted data, we replace the sigmoid function with polynomial approximations that are compatible with practical HE schemes.

Actually, the Taylor polynomials $T_d(x) = \sum_{k=0}^{d} (f^{(x)}(0)/k!)x^k$ have been commonly used for approximation of the sigmoid function [32–34]:

$$\sigma(x) = \frac{1}{2} + \frac{1}{4}x - \frac{1}{48}x^3 + \frac{1}{480}x^5 + O\left(x^7\right). \tag{3}$$

However, we observe that the size of error grows rapidly as $|x|$ increases. Besides, in order to guarantee the accuracy of the BP network, we have to use a higher degree Taylor polynomial, but it requires too many homomorphic

Table 3: Research works in secure machine learning.

Setting	Prior work	Problem	Activation	Technique
PaaS	DeepSecure [29]	DNN	ReLU, ReLU, sigmoid	MPC
	Gazelle [27]	CNN	ReLU	MPC, HE
	CryptoNets [30]	CNN	Square function	SHE
	FHE-DiNN [26]	DiNN	Sign	FHE
	Chameleon [28]	DNN	ReLU, sigmoid	MPC, HE
TaaS	SecureML [27]	LR, NN	Sigmoid, Softmax	MPC
	[31]	LR	Least squares approximation	HE
	Ours	BP	$L_3(x)$	HE

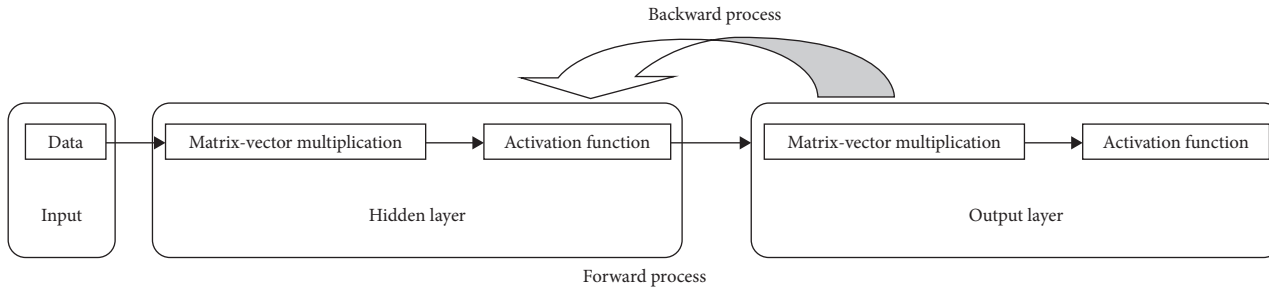

Figure 1: A conventional BP network with three layers.

(1) Set the number of iterations, weight matrix W, and bias vector b to small random initial values.
(2) Repeat
(3) Forward phase: beginning with the input nodes, compute weighted sums, and activation function for all nodes;
(4) Backward phase: compute gradients for all nodes starting from output nodes;
(5) Adjust the weights and biases
(6) until the number of iterations reaches a preset value.

Algorithm 1: The conventional BP training algorithm.

multiplications to be practically implemented. In summary, although Taylor expansions are more convenient and easier to compute, the accuracy of estimation is not always consistent because it is a local approximation near a certain point. Therefore, we introduce another good candidate for approximation with better approximation ability to replace the sigmoid function: optimal and uniform polynomial approximation of $\sigma(x)$. Not exactly, we find a polynomial function $L_n(x)$ that minimizes the absolute value of the error between $\sigma(x)$ and $L_n(x)$ within a given interval.

The Chebyshev polynomials are used to construct the optimal uniform approximation polynomials $L_n(x)$. The Chebyshev polynomials $C_n(x)$ can be simply defined as for $-1 \le x \le 1$,

$$C_n(x) = \cos(n \cdot \arccos x), \quad n = 0, 1, 2, \ldots \tag{4}$$

From the abovementioned definition, we can get two important properties of Chebyshev polynomials. The first property is that we can get a recurrence of Chebyshev polynomials $C_0(x) = 1$, $C_1(x) = x$, $C_{n+1}(x) = 2xC_n(x) - C_{n-1}(x)$. The second is that the Chebyshev polynomial $C_n(x)$ has n different zero points on the interval $[-1, 1]$, i.e., $x_k = \cos((2k-1)\pi/2n)$, $k = 1, 2, \ldots, n$. Then, we can get the important theorem in polynomial approximation as follows.

Theorem 1. *Let $f(x)$ be a continuous differentiable function on interval $[-1, 1]$, $L_n(x)$ be the interpolation polynomial, and its interpolation nodes $x_0, \cdots, x_n$ are the zero points of Chebychev polynomial $C_{n+1}(x)$, then $L_n(x)$ is the optimal and uniform polynomial approximation of $f(x)$ on interval $[-1, 1]$, and*

$$\max_{-1 \le x \le 1} \left| f(x) - L_n(x) \right| \le \frac{1}{2^n (n+1)!} \left\| f^{(n+1)}(x) \right\|_\infty. \tag{5}$$

Proof

$$\begin{aligned}\max_{-1\le x\le 1}\left|f(x)-L_n(x)\right| &\le \frac{1}{(n+1)!}\left\|f^{(n+1)}(x)\right\|_\infty \left\|(x-x_0)(x-x_1)\dots(x-x_n)\right\|_\infty \\ &= \frac{1}{(n+1)!}\left\|f^{(n+1)}(x)\right\|_\infty \left\|\frac{1}{2^n}C_{n+1}\right\|_\infty \\ &\le \frac{1}{2^n(n+1)!}\left\|f^{(n+1)}(x)\right\|_\infty .\end{aligned} \tag{6}$$

Therefore, it can be seen from the abovementioned theorem that to find the optimal uniform approximation polynomial of $f(x)$ on the interval $[-1,1]$, we only need to set the interpolation node of $L_n(x)$ as the zero point of Chebychev polynomial $C_{n+1}(x)$. For the function $f(x)$ on an interval $[a,b]$, we can take the transformation $x=((a+b/2)+(b-a/2))t,\ -1\le t\le 1$, so that $f(x)=f(((a+b/2)+(b-a/2))t)=g(t)$. Then, we can apply Theorem 1 to $g(t)$. We note that compared with Taylor polynomials, this method of polynomial approximation causes more similarities of derivatives with the sigmoid function, which might help produce a better model (see Figure 2).

To justify our claims, we compare the accuracy of the produced BP neural network model using different activation functions with the Iris dataset (see Table 4). □

3.2. Our SecureBP Network Model. In this section, we explain how to perform the lightweight interactive protocol to refresh ciphertexts during the training phase. To be precise, we explicitly describe a full pipeline of the evaluation of the SecureBP. We adopt the same assumptions as in the previous section so that the whole database can be encrypted in m ciphertexts.

First of all, in the setup phase, the user encrypts the dataset and sends them to the public cloud. The cloud randomly initializes weights and biases (in the initialization phase, the weights and biases can be plaintexts). Next, we introduce the iterative computing phase carried out in the cloud. The goal of each iteration is to update the weights and biases. Note that $ct.x_i$ ($ct.h_j$, $ct.o_k$, $ct.W^h_{j,i}$, $ct.W^o_{k,j}$, $ct.b^h_j$, $ct.b^o_k$) denotes the ciphertext of x_i (h_j, o_k, $W^h_{j,i}$, $W^o_{k,j}$, b^h_j, b^o_k, respectively). Each iteration consists of the following six steps:

Step 1. Cloud starts the iterative computation (here, $ct.r_j$ (including $ct.r_k$ in (8)) represents the encryption of a small random number, which has no effect on the correctness of the decryption):

$$\begin{aligned} ct.h'_j &= \sum_i ct.W^h_{j,i}\otimes ct.x_i \oplus ct.b^h_j, \\ ct.\widetilde{h}_j &= L_3\left(ct.h'_j\right)\oplus ct.r_j. \end{aligned} \tag{7}$$

Step 2. Cloud sends $\left\{ct.\widetilde{h}_j\right\}_{j\in[z]}$ to the user. After decrypting and reencrypting them, the user sends the refresh ciphertext $\left\{ct.h_j\right\}_{j\in[z]}$ to the cloud for further computation.

Step 3. Cloud computes

$$\begin{aligned} ct.o'_k &= \sum_i ct.W^o_{k,j}\otimes ct.h_i \oplus ct.b^o_k, \\ ct.\widetilde{o}_k &= L_3\left(ct.o'_k\right)\oplus ct.r_k. \end{aligned} \tag{8}$$

Step 4. Cloud sends $\{ct.\widetilde{o}_k\}_{k\in[d]}$ to the user. After decrypting and reencrypting them, the user sends the refresh ciphertext $\{ct.o_k\}_{k\in[d]}$ to the cloud for further computation.

Step 5. Cloud updates $\left\{ct.W^o_{k,j}\right\}$ and $\{ct.b^o_k\}$:

$$\begin{aligned} \mathrm{Err}^o_k &= ct.o_k\otimes\left(1-ct.o_k\right)\otimes\left(t_k-ct.o_k\right), \\ ct.W^o_{k,j} &= ct.W^o_{k,j}\oplus l\odot\mathrm{Err}^o_k\otimes ct.h_j, \\ ct.b^o_k &= ct.b^o_k\oplus l\odot\mathrm{Err}^o_k. \end{aligned} \tag{9}$$

Step 6. Cloud updates $ct.W^h_{j,i}$, $ct.b^h_j$,

$$\begin{aligned} \mathrm{Err}^h_j &= ct.h_j\otimes\left(1-ct.h_j\right)\left(\sum_k \mathrm{Err}^o_k\, ct.W^o_{k,j}\right), \\ ct.W^h_{j,i} &= ct.W^h_{j,i}\oplus l\odot\mathrm{Err}^h_j\otimes ct.x_i, \\ ct.b^h_j &= ct.b^h_j\oplus l\odot\mathrm{Err}^h_j. \end{aligned} \tag{10}$$

In the abovementioned iteration, we choose the interaction between the cloud and the user to avoid high-cost bootstrapping. We will send the outputs of the hidden layer and the output layer to the user. After the user refreshes these ciphertexts, they will be sent to the cloud to continue the subsequent homomorphic operations. Because the outputs of the hidden layer and output layer are two ciphertext vectors, with a total of $(z+d)$ ciphertexts, the communication cost between the cloud and the user is not high. Through the analysis of noise in the later section, we can find that the advantage of this interactive protocol makes the noise of ciphertext in the process of homomorphic operations grow linearly after it reaches a certain value (i.e., e^{33}).

In this process, it should also be noted that what the cloud sends to the user is not the true outputs of the hidden layer and output layer (i.e., $\{ct.h'_j\}_{j\in[z]}$ and $\{ct.o'_k\}_{k\in[d]}$), but the disturbed $\left\{ct.\widetilde{h}_j\right\}_{j\in[z]}$ and $\{ct.\widetilde{o}_k\}_{k\in[d]}$. The idea is to prevent the user from snooping into the cloud to train the neural network.

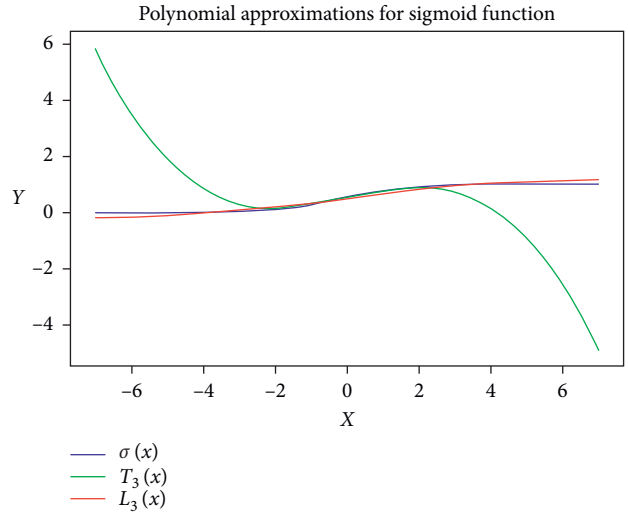

Figure 2: Polynomial approximation for the sigmoid function $\sigma(x)$ on the interval $[-7, 7]$. $L_3(x) = 0.500781 + 0.14670403x + 0.001198x^2 - 0.001006x^3$ generated by our method and $T_3(x) = (1/2) + (1/4)x - (1/4)x - (1/48)x^3$ generated by Taylor expansions.

Table 4: Accuracy of different activation functions on the Iris dataset.

Iteration	50 (%)	100 (%)	150 (%)	180 (%)	200 (%)	220 (%)	240 (%)	260 (%)	280 (%)	300 (%)
$\sigma(x)$	33.33	33.33	40.67	74.67	83.33	84.00	84.00	84.67	82.67	85.33
$T_3(x)$	33.33	33.33	40.67	51.33	81.33	84.00	84.00	84.00	82.67	82.67
$L_3(x)$	33.33	33.33	45.33	54.67	58.00	88.00	90.00	90.00	91.30	91.30

4. Estimation

In this section, we show the parameters setting for BP and the CKKS scheme and analyze the estimation and implementation results.

4.1. Parameters Setting and Estimation Results

4.1.1. Parameters for the BP Algorithm. In the BP model, the numbers of input nodes and output nodes are determined, while the number of hidden nodes is uncertain. In fact, the number of hidden nodes has an impact on the performance of the neural network; an empirical formula can determine the number of hidden nodes as follows:

$$z = \sqrt{d + m} + a, \tag{11}$$

where z is the number of hidden nodes, d is the number of input nodes, m is the output nodes, and a is an adjustment constant between 0 and 10.

Weights are initialized as uniformly random values in the range $[-0.1, 0.1]$. Feature values in each dataset are normalized between 0 and 1. The architecture and training parameters used in our secure neural network model are shown in Table 5, and we choose Iris, Diabetes, and Sonar datasets, which are from the University of California at Irvine (UCI) dataset repository [25]. The conventional BP learning network has the same parameters as the SecureBP algorithm.

4.1.2. Parameters for the CKKS Scheme. In the CKKS scheme, the coefficients of error polynomials are sampled from the discrete Gaussian distribution of standard deviation $\sigma = 3.2$ and a secret key is chosen randomly from the set of signed binary polynomials with the Hamming weight $h = 64$. We used the estimator of Albrecht et al. [35] to guarantee that the proposed parameter sets achieve at least 80 bit security level against the known attacks against the LWE problem.

We analyze the growth of noises in some ciphertexts, and Table 6 provides theoretical upper bounds on the noise growth during homomorphic operations. Note that e denotes the noise of a fresh ciphertext.

As can be seen from Table 6, the maximum size of growth noise during homomorphic operations is $\widetilde{O}(e^{33})$ [36]; we choose parameters as follows: $L = 10$, $N = 2^{15}$, $\log q = 55$, and $[\log Q_L] = 611$.

4.2. Estimation and Implementation Results. By carefully analyzing our SecureBP protocol, we calculate the number of homomorphic operations required for each step in the course of an iteration (as shown in Table 7).

Table 5: Datasets and parameters in BP.

Dataset	Number of samples	Number of features	Number of hidden nodes	Learning rate
Iris	150	4	4	0.6
Diabetes	768	8	5	0.5
Sonar	208	60	9	0.4

Table 6: Noise growth during the SecureBP process.

The ciphertext	The 1st iteration	The i-th iteration $(i = 2, 3, 4, \ldots)$
$ct.h_j$	$e^6 + O(e^5)$	$((i(i-1))/2)^3 \cdot e^{33} + O(e^{32})$
$ct.o_i$	$e^6 + O(e^5)$	$(i-1)^3 \cdot e^{15} + O(e^{14})$
$ct.W^o_{j,i}$	$e^4 + O(e^3)$	$i \cdot e^4 + O(e^3)$
$ct.W^h_{k,j}$	$e^{10} + O(e^9)$	$((i \cdot (i+1))/2) \cdot e^{10} + O(e^9)$

Table 7: Number of operations in SecureBP.

Step	Enc	Dec	Mult	CMult	Add
1	0	0	$mz + 2z$	0	$mz + z$
2	z	z	0	0	0
3	0	0	$zd + 2d$	0	$zd + d$
4	d	d	0	0	0
5	0	0	$3d$	$4d$	$5d$
6	0	0	$(d+3)z$	$3z$	$(d+2)z$
Total	$z + d$	$z + d$	$mz + 2zd + 5(d + z)$	$4d + 3z$	$mz + 2zd + 6d + 3zd$

Table 8: Homomorphic training of SecureBP.

Dataset	Enc (ms)	Dec (ms)	Mult (ms)	CMult (ms)	Add (ms)	Total (ms)
Iris	522	39	6068	836	167.7	7632.7
Diabetes	812	46.8	7872	1012	219.3	9962.1
Sonar	4060	78	20992	1716	580.5	2.0993×10^8

Table 9: Comparison of encrypted/unencrypted BP algorithm.

Dataset	Number of iterations	Error rate of SecureBP (%)	Error rate of conventional BP (%)
Iris	10	66.67	66.67
	23	20.40	16.67
Diabetes	10	36.97	34.71
	23	34.90	33.89
Sonar	10	21.45	18.26
	23	17.77	17.21

From Table 7, we can see that the computation time required for an iteration is only related to the number of nodes in each layer. Combined with the time required for each homomorphic operation in [36], we give the estimation time (Table 8) of training SecureBP network homomorphically with Iris, Diabetes, and Sonar datasets, and Table 9 shows the accuracy comparison of encrypted and unencrypted BP networks in the case of 10 and 23 iterations, respectively.

4.3. Efficiency and Accuracy Discussion. There are still some limitations in the application of our evaluation model to an arbitrary dataset. On the one hand, the HE system is a promising solution for the privacy issue, but its efficiency in real applications remains an open question. In other words, one constraint in our approach is that the efficiency of the SecureBP network is limited by the efficiency of homomorphic operations. On the other hand, we find that the accuracy of the network model is positively correlated with

the degree of approximate polynomials. However, the higher degree of polynomial means the more homomorphic operations, the more time it takes to train the network model. Therefore, we need the tradeoff between the training efficiency and accuracy of the model.

5. Conclusion and Future Work

In this paper, we present a SecureBP network model for homomorphic training. We introduce two methods, more accuracy polynomial approximation and lightweight interactive protocol, to solve the difficulties encountered when the CKKS scheme is used to protect the BP network, and our method has a good experimental performance on different datasets. For future work, we plan to explore how to train the deep neural network and the convolutional neural network effectively on encrypted data in a training-as-a-service setting.

Data Availability

All types of data used to support the findings of this study have been deposited in the University of California at Irvine (UCI) Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets.html).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported the National Natural Science Foundation of China under Grant no. 61672030.

References

[1] A. Esteva, B. Kuprel, R. A. Novoa et al., "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[2] V. Gulshan, L. Peng, M. Coram et al., "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *JAMA*, vol. 316, no. 22, pp. 2402–2410, 2016.

[3] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: a unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, Boston, MA, USA, June 2015.

[4] M. Barni, C. Orlandi, and A. Piva, "A privacy-preserving protocol for neural-network based computation," in *Proceedings of the 8th Workshop on Multimedia and Security*, pp. 146–151, ACM, Geneva, Switzerland, 2006.

[5] T. Chen and S. Zhong, "Privacy-preserving backpropagation neural network learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 10, pp. 1554–1564, 2009.

[6] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing via homomorphic encryption," *EURASIP Journal on Information Security*, vol. 2007, no. 1, Article ID 037343, 2007.

[7] A. Piva, C. Orlandi, M. Caini, T. Bianchi, and M. Barni, "Enhancing privacy in remote data classification," in *IFIP International Information Security Conference*, pp. 33–46, Springer, Boston, MA, USA, 2008.

[8] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, pp. 169–180, Academia Press, Ghent, Belgium, 1978.

[9] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Symposium on Theory of Computing-Stoc'09*, vol. 9, Bethesda, MY, USA, June 2009.

[10] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory*, vol. 6, no. 3, pp. 1–36, 2014.

[11] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[12] J. H. Cheon and D. Stehl´e, "Fully homomophic encryption over the integers revisited," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 513–536, Springer, Sofia, Bulgaria, April 2015.

[13] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in Cryptology—Crypto 2012, Lecture Notes in Computer Science*, vol. 7417, pp. 850–867, Springer, Berlin, Germany, 2012.

[14] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology—CRYPTO 2013*, vol. 8042, pp. 75–92, Springer, Berlin, Germany, 2013.

[15] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 24–43, Springer, Monaco and Nice, France, May 2010, Advances in Cryptology-EUROCRYPT.

[16] S. Halevi and V. Shoup, "Algorithms in HElib," *Advances in Cryptology—CRYPTO*, Springer, in *Proceedings of the Annual Cryptology Conference*, pp. 554–571, Springer, Santa Barbara, CA, USA, August 2014.

[17] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library—SEAL v2.1," *Financial Cryptography and Data Security* in *Proceedings of the International Conference on Financial Cryptography and Data Security*, vol. 10323, Springer, Cham, Switzerland, pp. 3–18, 2017.

[18] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the Innovations in Theoretical Computer Science 2012*, pp. 309–325, Cambridge, MA, USA, January 2012.

[19] L. Ducas and D. Micciancio, "FHEW: bootstrapping homomorphic encryption in less than a second," in *Advances in Cryptology—EUROCRYPT 2015, Part I, Lecture Notes in Computer Science*, vol. 9056, pp. 617–640, Springer, Berlin, Germany, 2015.

[20] I. Chillotti, N. Gama, M. Georgieva, and M. Izabach`ene, "Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds," in *Advances in Cryptology—ASIACRYPT 2016, Part I*, pp. 3–33, Springer, Berlin, Germany, 2016.

[21] I. Chillotti, N. Gama, M. Georgieva, and M. Izabach`ene, "Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE," in *Advances in Cryptology—ASIACRYPT 2017, Part I*, vol. 10624, pp. 377–408, Springer, Cham, Switzerland, 2017.

[22] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Advances in Cryptology—EUROCRYPT 2018, Proceedings, Part I*, pp. 360–384, Springer, Cham, Switzerland, 2018.

[23] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology—ASIACRYPT 2017, Part I*, vol. 10624, pp. 409–437, Springer, Cham, Switzerland, 2017.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Technical Report, California Univ San Diego La Jolla Inst for Cognitive Science, San Diego, CA, USA, 1985.

[25] C. L. Blake, *UCI Repository of Machine Learning Databases*, University of California, Irvine, CA, USA, 1998, http://www.ics.uci.edu/mlearn/MLRepository.html.

[26] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Advances in Cryptology—CRYPTO 2018, Proceedings, Part III, Lecture Notes in Computer Science*, vol. 10993, pp. 483–512, Springer, Cham, Switzerland, 2018, https://search.crossref.org/?q=Fast+homomorphic+evaluation+of+deep+discretized+neural+networks.

[27] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: a low latency framework for secure neural network inference," in *Proceedings of the USENIX Security 2018*, pp. 1651–1669, Baltimore, MD, USA, 2018.

[28] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: a hybrid secure computation framework for machine learning applications," in *Proceedings of the Asia Conference on Computer and Communications Security*, pp. 707–721, Incheon, Korea, June 2018.

[29] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "DeepSecure: scalable provably-secure deep learning," in *Proceedings of the 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, June 2018.

[30] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the International Conference on Machine Learning*, pp. 201–210, Newyork, NY, USA, June 2016.

[31] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," *BMC Medical Genomics*, vol. 11, no. S4, 2018.

[32] J. W. Bos, K. Lauter, and M. Naehrig, "Private predictive analysis on encrypted medical data," *Journal of Biomedical Informatics*, vol. 50, pp. 234–243, 2014.

[33] P. Mohassel and Y. Zhang, "SecureML: a system for scalable privacypreserving machine learning," in *Proceedings of the Symposium on Security and Privacy*, pp. 19–38, San Jose, CA, USA, May 2017.

[34] J. Yuan and S. Yu, "Privacy preserving back-propagation neural network learning made practical with cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 212–221, 2014.

[35] M. R. 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.

[36] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *Selected Areas in Cryptography—SAC*, vol. 11349, pp. 347–368, Springer, Cham, Switzerland, 2018.