Privacy Preserving Deep Computation Model on Cloud for Big Data Feature Learning

Qingchen Zhang, Laurence T. Yang, and Zhikui Chen

Abstract—To improve the efficiency of big data feature learning, the paper proposes a privacy preserving deep computation model by offloading the expensive operations to the cloud. Privacy concerns become evident because there are a large number of private data by various applications in the smart city, such as sensitive data of governments or proprietary information of enterprises. To protect the private data, the proposed model uses the BGV encryption scheme to encrypt the private data and employs cloud servers to perform the high-order back-propagation algorithm on the encrypted data efficiently for deep computation model training. Furthermore, the proposed scheme approximates the Sigmoid function as a polynomial function to support the secure computation of the activation function with the BGV encryption. In our scheme, only the encryption operations and the decryption operations are performed by the client while all the computation tasks are performed on the cloud. Experimental results show that our scheme is improved by approximately 2.5 times in the training efficiency compared to the conventional deep computation model without disclosing the private data using the cloud computing including ten nodes. More importantly, our scheme is highly scalable by employing more cloud servers, which is particularly suitable for big data.

Index Terms—Smart city, big data, deep computation model, cloud computing, BGV encryption, BGN encryption, high-order back-propagation

1 Introduction

Developing the smart city is the key to improve the efficiency, reliability, and security of a traditional city [1]. Smart city consists of intelligent transportation, smart grid, intelligent security and so on. With the development of these fields, recent years have witnessed the remarkable growth of smart cities [2]. With the massive deployment of various mobile devices, such as sensors and RFID, data are being collected at unprecedentedly rate in the smart city [3]. Therefore, it is critical for smart city planning, monitoring and controlling to develop big data modeling and analytic technologies [4], [5]. As a fundamental technique of big data analytic, feature learning can discover the underlying structure of big data to provide intelligent decision for developing smart city systems [6], [7], [8]. However, the characteristics of big data, referring to large scale of data, different types of data, and the speed of streaming data, pose feature learning many significant challenges [9], [10]. To tackle these challenges, we proposed a deep computation model for learning features on big data effectively in the previous work. Owning to the huge amount of data in the smart and high computational complexity, the deep computation model finds it difficult to perform in real-time with limited computing power and memory storage. Although the performance of computers has been improved, it still falls behind the growth of the big data size. Thus, how to support the real-time deep computation model training for big data feature learning is one of the most challenging issues in the smart city.

Motivation. Today, cloud computing has come to play a vital role in big data modeling and analytic [11], [12]. It has been successfully applied in industrial products and commercial fields that take advantage of big data [13], [14]. For example, with cloud computing, Google offers a wide variety of real-time services such as real-time searching, real-time translation and voice recognition [15]. Cloud computing provides us with strong computing power and massive storage space [16], [17], [18]. Therefore, it is an effective method to improve the efficiency of training deep computation model for big data feature learning by offloading the expensive operations to the cloud [19], [20]. However, privacy concerns bring forward in the cloud computing since there exist a large number of private data collected from the smart city, such as population and economic information. These data may contain sensitive data of governments or proprietary information of the enterprises [21], [22], [23]. Once they are disclosed, people's lives and property will be seriously threatened. Especially in the smart city, disclosure of sensitive data is not only a privacy issue but of legal concerns according to privacy protection laws such as the Health Insurance Portability and Accountability Act (HIPAA). Therefore, this paper focuses on the privacy preserving deep computation model with the cloud computing.

Challenges. The privacy preserving deep computation model poses a number of issues and challenges, especially for big data feature learning by incorporating the computing of the cloud. We discuss the key challenges in three aspects as follows: (1) To protect the private data and intermediate results, it requires secure computation of various operations needed by the deep computation model, including

- *Q. Zhang and Z. Chen are with the School of Software Technology, Dalian University of Technology, Dalian, Liaoning 116620, China. E-mail: qingchen@mail.dlut.edu.cn, zkchen@dlut.edu.cn.*
- *L. T. Yang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and the Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. E-mail: ltyang@ieee.org.*

Manuscript received 15 Feb. 2015; revised 13 July 2015; accepted 11 Aug. 2015. Date of publication 18 Aug. 2015; date of current version 13 Apr. 2016. Recommended for acceptance by S. Hu, G. Betis, R. Ranjan, and L. Wang. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TC.2015.2470255

additions, multiplications, and the nonlinear Sigmoid function. (2) To improve the efficiency of deep computation model training and big data feature learning, it requires to choose the efficient full homomorphic encryption scheme according to the major operations of the algorithms in the privacy preserving deep computation model. (3) To produce the correct result on the ciphertexts using the full homomorphic encryption scheme, the Sigmoid function is required to approximate as a new function involving only addition operations and multiplication operations.

Contributions. In this paper, we propose a privacy preserving deep computation model based on homomorphic encryption. The proposed scheme improves the efficiency by offloading the expensive computation tasks on the cloud. Furthermore, the proposed scheme prevents the disclosure of the private using homomorphic encryption which has been successfully used for data mining and knowledge discovery such as decision trees [25], [26], Bayesian networks [27], [28], support vector machines [29], and k-means [30].

To support secure computation of various operations such as additions and multiplications required by the high-order back-propagation algorithm, the paper encrypts the private data using the BGV encryption scheme [31] that is the currently most efficient full homomorphic encryption scheme and supports simultaneously supports arbitrary number of addition operations and multiplication operations. However, BGV does not support the exponentiation operation over ciphertexts, resulting in the failure to the secure computation of Sigmoid function that is the activation function of the deep computation model. To address this problem, the paper utilizes Taylor theorem to approximate the Sigmoid function as a polynomial function involving only addition operations and multiplication operations so that it is suitable for the privacy preserving high-order back-propagation algorithm. The main idea of the proposed algorithm can be summarized as follows: the client first encrypts the private data with the system public key and then uploads the ciphertexts to the cloud; cloud servers then perform the high-order back-propagation algorithm over the ciphertexts and return the encrypted results to the client; the client decrypts the results with which it updates the parameters of deep computation model. During this process, cloud servers learn no private data of the client and intermediate results, so our proposed scheme is secure.

Our contributions can be summarized as follows:

- To improve the efficiency of deep computation model training, we offload the expensive operations to the cloud. Only the encryption operations and the decryption operations are performed by the client while all the computation tasks are performed on the cloud.
- By encrypting the input data using the BGV encryption scheme before uploading them to the cloud, the proposed algorithm protects the private data.
- Since the BGV encryption scheme does not support the exponentiation operation required by Sigmoid function, the paper utilizes Taylor theorem to approximate the Sigmoid function as a polynomial function involving only addition operations and multiplication operations so that it is suitable for the privacy preserving high-order back-propagation algorithm.

Experimental results on two representative classification datasets and two real smart city datasets show that our scheme can efficiently train deep computation model for big data feature learning by offloading the expensive operations to the cloud without disclosing the private data. More importantly, our scheme is of high scalability by employing more cloud servers, which is particularly suitable for big data.

The rest of the paper is organized as follows: Section 2 presents the preliminaries related to this paper. The privacy preserving high-order back-order algorithm based on the BGV encryption scheme is illustrated in Section 3. Section 4 evaluates our proposed scheme and Section 5 reviews related works on the privacy preserving neural networks learning. Finally, the whole paper is concluded in Section 6.

2 Preliminaries

In this section, we present the technique preliminaries used in our proposed scheme including the tensor auto-encoder (TAE) proposed by our previous work and the BGV encryption scheme.

2.1 Tensor Auto-Encoder (TAE)

Basic auto-encoders and their variants work in the vector space. Vectors cannot represent a large number of heterogeneous data that are prevalent in big data, making auto-encoders difficult to learn features of big data. Aiming at this problem, tensor auto-encoders uses the tensor-based model for big data representation to model the highly non-linear distribution of various heterogeneous data [32], [33]. Given a training sample, two tensors $X \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and $H \in R^{J_1 \times J_2 \times \cdots \times J_N}$ are denoted as the values of input layer nodes and the values of hidden layer nodes respectively. TAE maps the input values to the hidden values via an encoder function:

$$H = f_\theta(W^{(1)} \odot X + b^{(1)}). \tag{1}$$

Then, TAE maps the hidden values to the reconstruction Y via the decoder function:

$$Y = h_{W,b}(X) = g_\theta(W^{(2)} \odot H + b^{(2)}), \tag{2}$$

where, $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$ is the parameter set, both the encoder function and the decoder function adopt the Sigmoid function [34]: $s_f(x) = 1/(1+e^{-x})$, $\odot$ represents the multi-dot product of two tensors, i,e., an $N+1$-order tensor $W \in R^{\alpha \times I_1 \times I_2 \times \cdots \times I_N}$ with α sub-tensors and an N-order tensor $A \in R^{I_1 \times I_2 \times \cdots \times I_N}$, defined as:

$$H = W \odot A, \forall h_{j_1 j_2 \ldots j_n} \in H, h_{j_1 j_2 \ldots j_n} = W_\beta \bullet A \left(\beta = j_n + \sum_{i=1}^{N-1} (j_i - 1) \prod_{t=i+1}^{N} J_t\right). \tag{3}$$

To encourage the representation obtained from a training input x to capture as much as possible of the unknown distribution from heterogeneous data, the tensor distance [35] is used in the reconstruction error, yielding the objective function as follows:

$$J_{TAE}(\theta; x, y) = \tfrac{1}{2}(h_{W,b}(x) - y)^T G(h_{W,b}(x) - y), \tag{4}$$

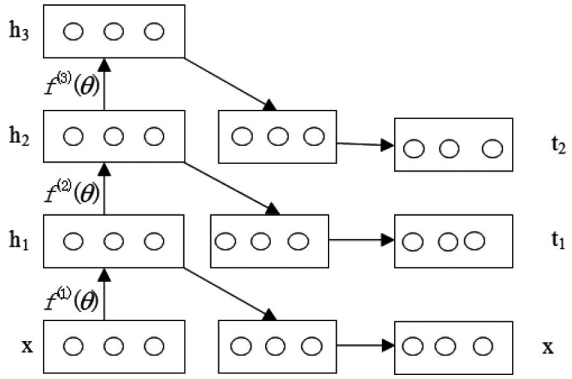


Fig. 1. Stacking tensor auto-encoders for pre-training.

where x, y is denoted as the vector form representation of the tensors X and Y.

For a training set $\{(X^{(1)}, Y^{(1)}), \ldots, (X^{(m)}, Y^{(m)})\}$ with m training examples, the reconstruction error of TAE is defined as:

$$J_{TAE}(\theta) = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}(h_{W,b}(x)-y)^T G(h_{W,b}(x)-y)\right)\right] + \frac{\lambda}{2}\left(\sum_{p=1}^{J_1\times\cdots\times J_N}\sum_{i_1=1}^{I_1}\cdots\sum_{i_N=1}^{I_N}\left(W^{(1)}_{pi_1\cdots i_n}\right)^2 + \sum_{q=1}^{I_1\times\cdots\times I_N}\sum_{j_1=1}^{J_1}\cdots\sum_{j_N=1}^{J_N}\left(W^{(2)}_{qj_1\cdots j_n}\right)^2\right). \quad (5)$$

In this reconstruction error, the first term is an average sum-of-squares error term and the second term is a regularization term, called a weight decay term, for decreasing the magnitude of the weights.

We use $z^{(2)}_{j_1j_2\ldots j_n}(1 \le j_i \le J_i, 1 \le i \le n), z^{(3)}_{i_1i_2\ldots i_n}(1 \le i_j \le I_j, 1 \le j \le n)$ to represent the input values of the hidden layer and the output layer respectively, and $a^{(2)}_{j_1j_2\ldots j_n}(1 \le j_i \le J_i, 1 \le i \le n), a^{(3)}_{i_1i_2\ldots i_n}(1 \le i_j \le I_j, 1 \le j \le n)$ to represent the activation values of the hidden layer and the output layer, respectively. To train the parameters of TAE, the high-order back-propagation algorithm has been proposed in our previous work outlined in Algorithm 1.

As outlined in Algorithm 1, high-order back-propagation is mainly composed of two stages: feed forward and error back-propagation. In the feed forward stage, the values at hidden layer and output layer are calculated using the parameters, the Sigmoid function, and the values at the previous layer. In the back propagation stage, the algorithm calculates the partial derivatives of the reconstruction error to the parameters for updating all the weights.

Several tensor auto-encoders can be stacked to a deep computation model for unsupervised feature learning on big data as shown in Fig. 1.

2.2 Homomoraphic Encryption Schemes

Homomorphic encryption was first introduced by Rivest, Adleman and Dertouzous shortly after the invention of RSA [36], [37]. Homomorphic encryption enables operations on plaintexts to be performed on their respective ciphertexts without disclosing the plaintexts [31]. Generally speaking, a homomorphic encryption scheme ε consists of four algorithms: $KeyGen$, $Encrypt$, $Decrypt$, and $Evaluate$ [38], [39].

$KeyGen$ takes a security parameter λ as input and produces a secret key sk and a public key pk, i.e., $KeyGen(\lambda) \rightarrow (pk, sk)$.

$Encrypt$ takes the public key pk and a plaintext m as input and produces the ciphertext c of m, i.e., $c \leftarrow Encrypt(m, pk)$.

$Decrypt$ takes the secret key sk and c as input and produces the plaintext m of c, i.e., $m \leftarrow Decrypt(c, sk)$.

$Evaluate$ takes the public key pk, a circuit C and a tuple of ciphertexts $(c_1, c_2, ..c_n)$ as input and produces the encrypted result c, i.e., $Decrypt(sk, c_1, c_2, ..c_n) = f(m_1, m_2, .., m_n)$, where f is the functionality that we want to perform.

Algorithm 1. High-order Back-propagation Algorithm.

Input: $\{(X^{(i)}, Y^{(i)})\}, iterater_{max}, \eta, threshold$
Output: $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$

for $iteration = 1, 2, \ldots, iterater_{max}$ **do**
  for $example = 1, 2, \ldots, N$ **do**
    for $j_1 = 1, 2, \ldots, J_1$ **do**
      $\cdots$;
      for $j_n = 1, 2, \ldots, J_N$ **do**
        $z^{(2)}_{j_1j_2\ldots j_n} = W^{(1)}_{\alpha} \cdot X + b^{(1)}_{j_1j_2\ldots j_n}$;
        $a^{(2)}_{j_1j_2\ldots j_n} = f(z^{(2)}_{j_1j_2\ldots j_n})$;
    for $i_1 = 1, 2, \ldots, I_1$ **do**
      $\cdots$;
      for $i_n = 1, 2, \ldots, I_N$ **do**
        $z^{(3)}_{i_1i_2\ldots i_n} = W^{(2)}_{\beta} \cdot a^{(2)} + b^{(2)}_{i_1i_2\ldots i_n}$;
        $h_{(i_1i_2\ldots i_n)W,b}(X) = a^{(3)}_{i_1i_2\ldots i_n} = f(z^{(3)}_{i_1i_2\ldots i_n})$;
    if $J_{TAE}(\theta) > threshold$ **then**
      for $i_n = 1, 2, \ldots, I_1 \times I_2 \times \cdots \times I_N$ **do**
        $\sigma^{(3)}_i = (a^{(3)}_i \cdot (1 - a^{(3)}_i)) \cdot \sum_{j=1}^{I_1\times I_2\times\cdots\times I_N} g_{ij}(a^{(3)}_j - y_j)$;
      for $j_1 = 1, 2, \ldots, J_1$ **do**
        $\cdots$;
        for $j_n = 1, 2, \ldots, J_N$ **do**
          $\sigma^{(2)}_{j_1j_2\ldots j_n} = (\sum_{i_1=1}^{I_1}\cdots\sum_{i_n}^{I_n} w^{(2)}_{\lambda j_1j_2\ldots j_n} \cdot \sigma^{(3)}_{i_1i_2\ldots i_n}) f'(z^{(2)}_{j_1j_2\ldots j_n})$;
      for $i_1 = 1, 2, \ldots, I_1$ **do**
        $\cdots$;
        for $i_n = 1, 2, \ldots, I_N$ **do**
          $\Delta b^{(2)}_{i_1i_2\ldots i_n} = \Delta b^{(2)}_{i_1i_2\ldots i_n} + \sigma^{(3)}_{i_1i_2\ldots i_n}$;
          for $j_1 = 1, 2, \ldots, J_1$ **do**
            $\cdots$;
            for $j_n = 1, 2, \ldots, J_N$ **do**
              $\Delta w^{(2)}_{i_1i_2\ldots i_nj_1j_2\ldots j_n} = \Delta w^{(2)}_{i_1i_2\ldots i_nj_1j_2\ldots j_n} + a^{(2)}_{j_1j_2\ldots j_n} \cdot \sigma^{(3)}_{i_1i_2\ldots i_n}$;
      for $j_1 = 1, 2, \ldots, J_1$ **do**
        $\cdots$;
        for $j_n = 1, 2, \ldots, J_N$ **do**
          $b^{(1)}_{j_1j_2\ldots j_n} = \Delta b^{(1)}_{j_1j_2\ldots j_n} + \sigma^{(2)}_{j_1j_2\ldots j_n}$;
          for $i_1 = 1, 2, \ldots, I_1$ **do**
            $\cdots$;
            for $i_n = 1, 2, \ldots, I_N$ **do**
              $\Delta w^{(1)}_{j_1j_2\ldots j_ni_1i_2\ldots i_n} = \Delta w^{(1)}_{j_1j_2\ldots j_ni_1i_2\ldots i_n} + x_{i_1i_2\ldots i_n} \cdot \sigma^{(2)}_{j_1j_2\ldots j_n}$;
        $W = W - \eta \times (\frac{1}{N}\Delta w)$;
        $b = b - \eta \times (\frac{1}{N}\Delta b)$;

Earlier homomorphic encryption schemes, called semi homomorphic encryption, only support single operation-either addition or multiplication. For example, some representative additively homomorphic encryption schemes are Okamoto-Uchiyama [40], Pallier [41] and Damard-Jurik [42] while RSA [37] and ElGamal [43] are representatives of the multiplicatively homomorphic encryption schemes. The first fully homomorphic encryption scheme using ideal lattices that allows both addition and multiplication to be implemented by Craig in 2009 [39]. Recently, many fully homomorphic encryption schemes have been proposed such as BGV [31], Bral2 [44] and GSW13 [45].

BGV is a LWE/RLWE-based leveled full homomorphic encryption scheme introduced by Brakerski et al. [31]. The BGV scheme begins a Setup procedure that chooses a μ-bit modulus q and the parameters: the dimension $n = n(\lambda, \mu)$, the degree $d = d(\lambda, \mu)$, the distribution $\chi = \chi(\lambda, \mu)$, and $N = \lceil (2n+1)\log q \rceil$ for given a security parameter.

To produce the correct result, Brakerski devised a key Switching Procedure and a modulus Switching Procedure in the BGV scheme. Specifically, the key Switching Procedure aims to reduce the dimension of the ciphertext by the following algorithms: $SwitchKeyGen(s_1 \in R_q^{n1}; s_2 \in R_q^{n2})$ and $SwitchKey(\tau_{s1 \to s2}, c_1)$. The former takes the two secret key vectors, the respective dimensions of these vectors, and the modulus q as input. The corresponding output is some auxiliary information $\tau_{s_1 \to s_2}$ that enables the switching. The latter takes this auxiliary information and a ciphertext encrypted under s_1 as input, and outputs a new ciphertext c_2. The modulus Switching Procedure is used to reduce the noises by designing an algorithm $Scale(c, q, p, r)$. We refer to [31] for details of the BGV scheme.

The BGV encryption scheme supports unlimited number of addition operations and multiplication operations without bootstrapping. Furthermore, BGV is more efficient than other full homomorphic encryption schemes. It is successfully used in many applications such as cloud computing and security computing. Therefore, the paper uses the BGV encryption scheme in the proposed privacy preserving deep computation model.

3 Privacy Preserving High-Order Back-Propagagtion Algorithm

In this section, we illustrate the proposed privacy preserving high-order back-propagation algorithm based on the BGV encryption scheme. We aim at training deep computation model efficiently by incorporating the computing power of the cloud without revealing private data. To achieve this goal, the main idea of our proposed scheme is to implement a privacy preserving equivalence for each step of the original high-order back-propagation algorithm outlined in Algorithm 1. We describe the operations of the BGV encryption scheme required by the proposed algorithm first, followed by the details.

3.1 Secure Operations of the BGV Encryption Scheme

To implement the privacy preserving high-order back-propagation algorithm, the secure operations of the BGV encryption scheme are needed, including encryption, decryption, secure additions and secure multiplications. Assuming that the secret key and the public key are $sk = (1, s'[1], s'[2], \ldots, s'[n]) \in R_q^{n+1}$ and $pk = A$, respectively for the parameters $params = (\mu, q, d, n, N, \chi)$ of the BGV encryption scheme, the secure operations are described as follows [31]:

1) *Encryption*: Given a message $m \in R_2$, encrypt it as: $c \leftarrow m + A^T r \in R_q^{n+1}$.
2) *Decryption*: Given a cihpertext c and the respective secret key s_j, decrypt it to get the plaintext $m \leftarrow ((< c, s_j > \mod q) \bmod 2)$.
3) *SecureAddition*: Given the ciphertexts c_1 and c_2 of messages m_1 and m_2, the cloud calculates their sum as: let $c_3 \leftarrow c_1 + c_2 \bmod q_j$, the sum of c_1 and c_2 is $c_4 \leftarrow \mathrm{R}efresh(c_3, \tau(s_j' \to s_{j-1}), q_j, q_{j-1})$.
4) *SecureProduct*: Given the ciphertexts c_1 and c_2 of messages m_1 and m_2, the cloud calculates their product as: let $c_3 \leftarrow c_1 \otimes c_2 \bmod q_j$, the product of c_1 and c_2 is $c_4 \leftarrow \mathrm{R}efresh(c_3, \tau(s_j' \to s_{j-1}), q_j, q_{j-1})$.

TABLE 1
Operations Used in the High-Order Back-Propagation Algorithm

Operation	Homomorphic	Example
$+$	yes	$\sum_{k=1}^{a} x_k \times w_{jk}^h$
$-$	yes	$a_j^{(3)} - y_j$
$\times$	yes	$a_{j_1 j_2 \cdots j_n}^{(2)} \times \sigma_{i_1 i_2 \ldots i_n}^{(3)}$
e^x	no	e^{-x}
$\div$	no	$1/(1+e^{-x})$

3.2 Approximation of Sigmoid Function

As described in Algorithm 1, the high-order back-propagation algorithm requires addition operations, subtraction operations, multiplication operations, division operantions, and exponentiation operations shown in the Table 1.

For the five types of operations used in the deep computation model training, i.e., addition $+$, subtraction $-$, multiplication $\times$, division $\div$, and exponentiation e^x, the first three operations are homomorphic, while the last two are non-homomorphic. The BGV encryption scheme does not support the division operation and exponentiation operation required by the Sigmoid function used as the activation function in the high-order back-propagation algorithm. Specially, the BGV encryption does not support the two operations over ciphertexts, i.e., calculating $C(1/x)$ and $C(e^x)$ for given $C(x)$. To support secure computation of the Sigmoid function, we remove the exponentiation operation and the division operation by using Taylor theorem to approximate the Sigmoid function to a polynomial function as follows:

$$\begin{aligned} y &= \frac{1}{1+e^{-x}} = \frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + o(x^4) \\ &\approx \frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} \approx 0.5 + 0.25x - 0.02x^3. \end{aligned} \tag{6}$$

As shown in the approximation of Sigmoid function in (6), the major challenge of secure computation of the equation becomes to calculate x^3. Since it can be calculated by

$x^3 = x \times x \times x$, the approximation of Sigmoid function in (7) involves only addition operations and multiplication operations. Therefore, the activation function can be calculated securely by Algorithm 2.

Algorithm 2. Secure Computation of Activation Function on Cloud.

Input: $C(x), C(0.5), C(0.25) and C(-0.02)$
Output: $C(y)$
1 Using secure addition to calculate: $C_1 = C(0.25) \times C(x)$;
2 Using secure multiplication to calculate: $C_2 = C(-0.02) \times C(x) \times C(x) \times C(x)$;
3 Using secure addition to calculate: $C(y) = C(0.5) + C_1 + C_2$;
4 **return** $C(y)$;

3.3 Scheme Overview

Given the initial parameters $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$, the task of the privacy preserving deep computation model on cloud is to train the parameters $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$ efficiently by performing the privacy preserving high-order back-propagation algorithm on the cloud without disclosing the private data.

To train the parameters $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$ of the deep computation model securely, the proposed scheme encrypts the training samples, i.e., input data $\{x_1, x_2, \ldots, x_a\}$, output data $\{t_1, t_2, \ldots, t_c\}$ and initialized parameters $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$ in the client and uploads the ciphertexts to the cloud, allowing the cloud servers to perform one iteration of the privacy preserving high-order back-propagation algorithm. The client downloads the results from the cloud and decrypts them for updating the parameters once, and then encrypts the updated parameters and uploads the encrypted parameters to the cloud for performing one more iteration of the privacy preserving high-order back-propagation algorithm. The repetitions will not be terminated until the error is within the threshold or the max number of iterations is exceeded. The overall scheme is outlined in Algorithm 3.

Algorithm 3. Overall Scheme of Privacy Preserving High-order Back-propagation Algorithm.

Input: $\{x_1, x_2, \ldots, x_a\}$, $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$, $iteration_{\max}$, η
Output: $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$
1 Client:;
2 Using encryption to encrypt the training samples;
3 Upload the encrypted training samples to the cloud;
4 Randomly initialize the parameters;
5 **for** $iteration = 1, 2, \ldots, iterater_{max}$ **do**
6 Using encryption to encrypt the parameters;
7 Upload the encrypted the parameters to the cloud;
8 Cloud:;
9 Perform Algorithms 4 and 5 over the ciphtexts;
10 Update the ciphertexts of $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$;
11 Send the results to the client;
12 Client:;
13 Using decryption operation to decrypt the results for updating the parameters;

In the proposed scheme, we encrypted the data and parameters using the BGV method which is a LWE/RLWE-based leveled full homomorphic encryption scheme. Therefore, when the number of the iteration is more than one, the depth of the circuit will increase rapidly with the increasing number of the iteration. The increase of the circuit depth will severely reduce the efficiency of the deep computation model training, and even produce the incorrect result. So, the parameters need be sent to the client for re-encryption after one iteration is performed.

As shown in the Algorithm 3, only the encryption operations and the decryption operations are performed by the client while all the computation tasks are performed on the cloud. So the proposed scheme can improve the training efficiency by incorporating the computing power of the cloud. The proposed scheme could not only protect the private data of users but also the parameters of the deep computation model since the client simultaneously encrypts the training data and the parameters. Furthermore, all the computation operations are performed on the ciphertexts in the cloud without disclosing private data. Therefore, our scheme is secure.

3.4 Privacy Preserving High-Order Back-Propagation Algorithm on Cloud

Cloud servers perform the privacy preserving high-order back-propagation algorithm over the ciphertexts for updating the parameters after receiving the encrypted data and parameters from the client. According to the Algorithm 1, cloud servers are required to complete the following computation tasks.

1) In the feed forward stage, the cloud is required to calculate the values of z^2, z^3, a^2, and a^3 over the respective ciphertexts. Since only addition operations and multiplication operations are required by the secure computation of the the values of z^2 and z^3, the cloud can complete this task using the secure addition operation and the secure multiplication operation of the BGV encryption scheme. Afterwards, the cloud can use Algorithm 2 to calculate the values of a^2 and a^3 which are the results of the activation function of z^2 and z^3, respectively. The feed forward stage is outlined in Algorithm 4.
2) The task in the back propagation stage is to securely calculate the values of σ^2, σ^3, ΔW^l, and Δb^l which requires only addition operations and multiplication operations, so the cloud can complete this task by using the secure addition operation and the secure multiplication operation of the BGV encryption scheme. The back propagation stage is outlined in Algorithm 5.

4 Performance Evaluation

To evaluate the performance of our cloud-based privacy-preserving deep computation model, we executed experiments on the cloud platform established in the Laboratory of Computer Architecture and Cloud Computing, including 10 nodes with 3.2 GHz Core i7 CPU and 4 GB memory. In Section 4.1, we numerically evaluate the performance of our proposed scheme in terms of computation cost and communication cost. In Section 4.2, we evaluated the performance of our proposed scheme using two representative classification

datasets, namely STL-10 and NUS-WIDE [46], [47]. In Section 4.3, we applied our proposed scheme to two real datasets, namely PeMS and DLeMP, to evaluate our proposed scheme further [48], [49]. Finally, we evaluate the scalability of our proposed scheme in terms of speedup on the cloud.

Algorithm 4. Feed Forward of Privacy Preserving High-order Back-propagation Algorithm on Cloud.

Input: $\{(X^{(i)}, Y^{(i)})\}, \theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$
Output: $\{z^{(2)}, z^{(3)}; a^{(2)}, a^{(3)}\}$
for $iteration = 1, 2, \ldots, iterater_{max}$ **do**
  for $example = 1, 2, \ldots, N$ **do**
    for $j_1 = 1, 2, \ldots, J_1$ **do**
      $\cdots$;
      for $j_n = 1, 2, \ldots, J_N$ **do**
        // Using secure addition and multiplication to calculate $z^{(2)}_{j_1 j_2 \ldots j_n}$;
        $z^{(2)}_{j_1 j_2 \ldots j_n} = W^{(1)}_{\alpha} \cdot X + b^{(1)}_{j_1 j_2 \ldots j_n}$;
        // Using Algorithm 2 to calculate $a^{(2)}_{j_1 j_2 \cdots j_n}$;
        $a^{(2)}_{j_1 j_2 \ldots j_n} = f(z^{(2)}_{j_1 j_2 \ldots j_n})$;
    for $i_1 = 1, 2, \ldots, I_1$ **do**
      $\cdots$;
      for $i_n = 1, 2, \ldots, I_N$ **do**
        // Using secure addition and multiplication to calculate $z^{(3)}_{i_1 i_2 \ldots i_n}$;
        $z^{(3)}_{i_1 i_2 \ldots i_n} = W^{(2)}_{\beta} \cdot a^{(2)} + b^{(2)}_{i_1 i_2 \ldots i_n}$;
        // Using Algorithm 2 to calculate $h_{(i_1 i_2 \ldots i_n)W,b}(X)$;
        $h_{(i_1 i_2 \ldots i_n)W,b}(X) = a^{(3)}_{i_1 i_2 \ldots i_n} = f(z^{(3)}_{i_1 i_2 \ldots i_n})$;

4.1 Numerical Analysis

In this section, we evaluate the performance the proposed scheme in terms of computation cost and communication cost. To express clearly, the time cost of one addition operation, one multiplication operation and one modulus operation on Ring R are denoted by ADD, MUL and MOD, respectively, in the following part.

Computation cost. In the proposed scheme, the client needs to encrypt all its private data only once before uploading the encrypted data to the cloud. After the learning process starts, the client needs to encrypt all the parameters and decrypt all the intermediate results once in each iteration while the cloud needs to perform the Algorithm 4 and the Algorithm 5 once in each iteration.

For the training sample represented by a tensor of the form $R^{I_1 \times I_2 \times \cdots \times I_N}$, the client needs to encrypt the sample with $\prod_{i=1}^{N} I_i \times (n+1) \times N$ (ADD + MUL) using the encryption operation of the BGV encryption scheme with the parameter (μ, q, d, n, N, χ). Note that encrypting the training samples is the one-time cost performed before learning. For the tensor auto-encoder, the basic module of the deep computation model, with the configuration $\prod_{i=1}^{N} I_i - \prod_{j=1}^{N} J_j - \prod_{i=1}^{N} I_i$ and the parameter set $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$, the client needs to encrypt the parameters with $\prod_{i=1}^{N} J_i (2\prod_{j=1}^{N} I_j + 1) + \prod_{i=1}^{N} I_i) \times (n+1) \times N$ (ADD + MUL) and decrypt the intermediate results with $(\prod_{i=1}^{N} J_i (2\prod_{j=1}^{N} I_j + 1) + \prod_{i=1}^{N} I_i) \times (n+1)$ MUL, $(\prod_{i=1}^{N} J_i (2\prod_{j=1}^{N} I_j + 1) + \prod_{i=1}^{N} I_i) \times n$ ADD and $2(\prod_{i=1}^{N} J_i (2\prod_{j=1}^{N} I_j + 1) + \prod_{i=1}^{N} I_i)$ MOD in each iteration.

Algorithm 5. Back Propagation of Privacy Preserving High-order Back-propagation Algorithm on Cloud.

Input: $\{(X^{(i)}, Y^{(i)})\}, \theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}, \{z^{(2)}, z^{(3)}; a^{(2)}, a^{(3)}\}, \eta$
Output: $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$
for $iteration = 1, 2, \ldots, iterater_{max}$ **do**
  for $example = 1, 2, \ldots, N$ **do**
    for $i_n = 1, 2, \ldots, I_1 \times I_2 \times \cdots \times I_N$ **do**
      // Using secure addition and multiplication to calculate $\sigma^{(3)}_i$;
      $\sigma^{(3)}_i = (a^{(3)}_i \cdot (1 - a^{(3)}_i)) \cdot \sum_{j=1}^{I_1 \times I_2 \times \cdots \times I_N} g_{ij}(a^{(3)}_j - y_j)$;
    for $j_1 = 1, 2, \ldots, J_1$ **do**
      $\cdots$;
      for $j_n = 1, 2, \ldots, J_N$ **do**
        // Using secure addition and multiplication to calculate $\sigma^{(2)}_{j_1 j_2 \ldots j_n}$;
        $\sigma^{(2)}_{j_1 j_2 \ldots j_n} = f'(z^{(2)}_{j_1 j_2 \ldots j_n}) \cdot (\sum_{i_1=1}^{I_1} \cdots \sum_{i_n}^{I_n} w^{(2)}_{\lambda j_1 j_2 \ldots j_n} \cdot \sigma^{(3)}_{i_1 i_2 \ldots i_n})$;
    for $i_1 = 1, 2, \ldots, I_1$ **do**
      $\cdots$;
      for $i_n = 1, 2, \ldots, I_N$ **do**
        // Using secure addition to calculate $\Delta b^{(2)}_{i_1 i_2 \ldots i_n}$;
        $\Delta b^{(2)}_{i_1 i_2 \ldots i_n} = \Delta b^{(2)}_{i_1 i_2 \ldots i_n} + \sigma^{(3)}_{i_1 i_2 \ldots i_n}$;
        for $j_1 = 1, 2, \ldots, J_1$ **do**
          $\cdots$;
          for $j_n = 1, 2, \ldots, J_N$ **do**
            // Using secure addition and multiplication to calculate $\Delta w^{(2)}_{i_1 i_2 \ldots i_n j_1 j_2 \ldots j_n}$;
            $\Delta w^{(2)}_{i_1 i_2 \ldots i_n j_1 j_2 \ldots j_n} = \Delta w^{(2)}_{i_1 i_2 \ldots i_n j_1 j_2 \ldots j_n} + a^{(2)}_{j_1 j_2 \ldots j_n} \cdot \sigma^{(3)}_{i_1 i_2 \ldots i_n}$;
    for $j_1 = 1, 2, \ldots, J_1$ **do**
      $\cdots$;
      for $j_n = 1, 2, \ldots, J_N$ **do**
        // Using secure addition to calculate $b^{(1)}_{j_1 j_2 \ldots j_n}$;
        $b^{(1)}_{j_1 j_2 \ldots j_n} = \Delta b^{(1)}_{j_1 j_2 \ldots j_n} + \sigma^{(2)}_{j_1 j_2 \ldots j_n}$;
        for $i_1 = 1, 2, \ldots, I_1$ **do**
          $\cdots$;
          for $i_n = 1, 2, \ldots, I_N$ **do**
            // Using secure addition and multiplication to calculate $\Delta w^{(1)}_{j_1 j_2 \ldots j_n i_1 i_2 \ldots i_n}$;
            $\Delta w^{(1)}_{j_1 j_2 \ldots j_n i_1 i_2 \ldots i_n} = \Delta w^{(1)}_{j_1 j_2 \ldots j_n i_1 i_2 \ldots i_n} + x_{i_1 i_2 \ldots i_n} \cdot \sigma^{(2)}_{j_1 j_2 \ldots j_n}$;

In the feed forward stage, by using Algorithms 2 and 4, the cloud performs $(n+1)((\prod_{i=1}^{N} I_i + \prod_{j=1}^{N} J_j)(n+1+2N) + 12N + 8n + 4)$ MUL, $2(n+2)(\prod_{i=1}^{N} I_i + \prod_{j=1}^{N} J_j + 6)N$ ADD and $(n+1)((\prod_{i=1}^{N} I_i + \prod_{j=1}^{N} J_j)(n+2) + 8n + 12)$ MOD to calculate the values of every hidden layer node and output layer node. In the back-propagation stage, to get the residual error values of every hidden layer node and output layer node, the cloud needs to perform $(n+1)((\prod_{i=1}^{N} I_i + \prod_{j=1}^{N} J_j)(3N + n + 1) + 4N + 4n + 2)$ MUL, $(n+2)(3(\prod_{i=1}^{N} I_i + \prod_{j=1}^{N} J_j) + 4)N$ ADD and $(n+1)((\prod_{i=1}^{N} I_i + \prod_{j=1}^{N} J_j)(n+3) + 4n + 2)$ MOD. Then, by using the

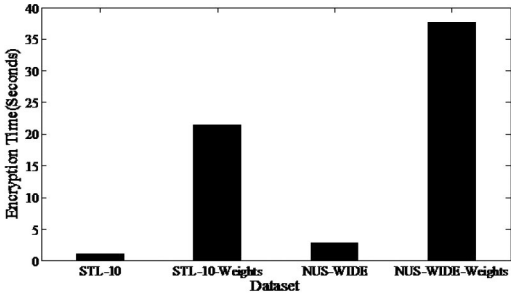


Fig. 2. Encryption time of datasets and weights.

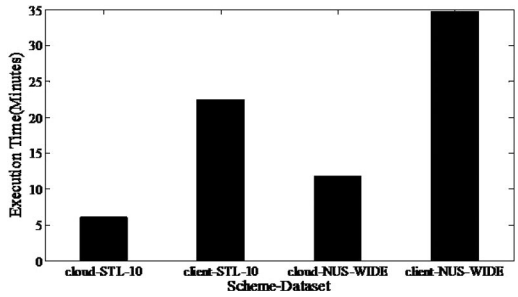


Fig. 3. Execution time of one iteration.

Algorithm 5, the cloud performs $2\prod_{i=1}^{N} I_i \prod_{j=1}^{N} J_j(n+1)$ $(n+1+3N)$ MUL, $2N(\prod_{i=1}^{N} I_i \prod_{j=1}^{N} J_j + 2)$ ADD and $2(\prod_{i=1}^{N} I_i \prod_{j=1}^{N} J_j + 1)(n+1)(n+3)$ MOD for the increments of the parameters. Although the computation cost on the cloud will increase with the increase of the training samples and the number of the sample attributes, cloud can handle it in parallel efficiently.

Communication cost. Before the learning process starts, the client needs to send $m \times \prod_{i=1}^{N} I_i$ messages with $m \times \prod_{i=1}^{N} I_i \times (n+1) \times \mu$ bits, where m is the number of the training samples to the cloud. Then, the client needs to exchange $\prod_{i=1}^{N} J_i(2\prod_{j=1}^{N} I_j + 1) + \prod_{i=1}^{N} I_i$ messages with $\prod_{i=1}^{N} J_i(2\prod_{j=1}^{N} I_j + 1) + \prod_{i=1}^{N} I_i \times (n+1) \times \mu$ bits with the cloud during the one round privacy preserving high-order back-propagation learning process.

From the above analysis, by offloading all the computation tasks to the cloud, the client only needs to perform the encryption/decryption operations and the cloud can perform most of the computation tasks in parallel for improving the efficiency. Furthermore, compared to the computation tasks, the computational cost is so small that it can be almost negligible.

4.2 Performance Analysis on Classification Datasets

In this section, we provide the experimental results of the privacy preserving high-order back-propagation algorithm on cloud and evaluate the performance of the proposed algorithm in terms of data encryption time, computational cost and the accuracy. The datasets are described first, followed by the experimental results.

In the STL-10 dataset, there are 500 training images grouped into 10 categories. We collected 10,000 images grouped into 20 categories from the NUS-WIDE dataset as the training set. The learning parameters in our experiments are set: $iteration_{max} = 100$ and $\eta = 0.03$.

Data encryption time. Before uploading the private data to cloud for the privacy preserving high-order back-propagation algorithm, the client needs to encrypt the input data and the parameters of the deep computation model with the encryption operation of the BGV encryption scheme. Fig. 2 shows the data encryption time of the two training sets and their respective weights, varying from 2.32 to 38.3 s. The data encryption time is greatly affected by the number of samples and features. Generally speaking, the encryption time will increase with the growth of the data size. For the training samples, the encryption time is a one-time cost and can be precomputed offline while the parameters are encrypted once in each iteration during the learning process.

Computation cost. To evaluate the efficient performance, we perform the privacy preserving high-order back-propagation algorithm on the cloud while performing the conventional high-order back-propagation algorithm on the client. Fig. 3 shows the execution time of two schemes for one iteration.

As shown in Fig. 3, our scheme costs only about 28 percent learning time compared to the conventional high-order back-propagation algorithm for one iteration. In other words, the efficiency of the proposed scheme is 2.5 times higher than that of the conventional high-order back-propagation algorithm for one iteration.

Fig. 4 shows the overall time required by performing the two algorithms on the two training sets. The execution time of the proposed scheme is about 34 percent of the conventional high-order back-propagation algorithm. For the overall execution time, the efficiency of the proposed scheme is two times higher than that of the conventional high-order back-propagation algorithm rather than 2.5 times. This is because the proposed scheme cost the addition overhead for the data encryption/decryption operations and data communication between the client and the cloud. In particular, the client needs to encrypt the parameters and decrypt the intermediate results once in each iteration and to communicate with the cloud for uploading the encrypted parameters and downloading the intermediate results.

Accuracy analysis. To analyze the accuracy loss in our privacy preserving high-order back-propagation algorithm, we classify the two datasets using the parameters trained by the proposed scheme and compare it with the non-privacy preserving high-order back-propagation algorithm. The classification accuracy is defined as CN/N, where CN represents the number of the objects that are classified correctly and N denotes the total number of the objects.

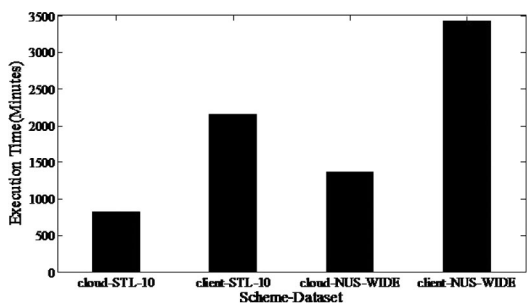


Fig. 4. Execution time of two algorithms.

TABLE 2
Classification Accuracy

Dataset	C-DCM(%)	PP-DCM(%)
STL-10	87.2	85.5
NUS-WIDE	81.7	79.8

Table 2 shows the average classification accuracy of the privacy preserving deep computation model (PP-DCM) and the conventional deep computation model (C-DCM).

As shown in the Table 2, our scheme achieves a lower accuracy performance than the non-privacy-preserving deep computation model due to the approximation of the activation function. Specially, our scheme produces about 2 percent more error than the conventional deep computation model when the number of Taylor series terms was set as 3.

Next, we reduce the accuracy loss by extending the number of Taylor series terms for 3 to 13. Fig. 5 shows the classification accuracy with regard to the different number of Taylor series terms.

According to Fig. 5, as the number of Taylor series terms grows, the classification accuracy increases. In other words, adding more Taylor series terms can reduce the accuracy lost effectively. Specifically, the accuracy lost can be reduced by about one percent by extending the number of Taylor series terms from 3 to 9. However, when the number of Taylor series terms is more than 9, the accuracy is improved very slightly with the increasing number of Taylor series terms. We use only three Taylor series terms rather than nine or more series terms in this paper because adding more series terms will increase the levels of the BGV encryption scheme and lower the performance of the privacy preserving high-order back-propagation algorithm.

4.3 Performance Analysis on Prediction Datasets

Intelligent transportation and wisdom economy are two important parts of the smart city. In this section, we use two real datasets, namely Performance Measurement System (PeMS) and Dalian Economy Management Platform (DLeMP), to evaluate the performance of the proposed scheme in practical applications.

PeMS is the most widely used dataset in traffic flow prediction [55]. The data are continuously collected from more than 8,100 freeway locations over 39,000 individual detectors deployed statewide throughout the State of California [56]. To evaluate the performance of our scheme, we aggregated the collected data by different detectors to get the average traffic flow of the freeway with many detectors. In this paper, we chose the collected traffic flow data of the first 10 months in 2014 as the training set and the later two months as the testing set. DLeMP is the commonly used economy prediction dataset, collected from more than 300 towns and streets of Dalian in the past 30 years. The dataset consists of 500,000 items, each with 52 attributes. To evaluate the proposed scheme, we selected 26 representative attributes to predict the gross economic product (GDP) of Dalian. In this experiment, the data of the first 20 years was used as the training set and the later 10 years as the testing set. The learning parameters in our experiments are set: $iterater_{max} = 50, timeintervals = 18, layersize = 3$ and $\eta = 0.01$, which were proved to produce the best results.

In this section, we present the experimental results of our proposed scheme on the PeMS dataset and the DLeMP dataset in terms of computation cost and prediction accuracy. At the same time, we compare the proposed scheme with the conventional deep computation model.

Computational cost. To evaluate the efficient performance of our scheme, we perform privacy preserving high-order back-propagation algorithm on the cloud while performing the non-privacy preserving high-order back-propagation algorithm on the client. In this experiment, we use 1/5, 2/5, 3/5, 4/5 and all the samples in the two datasets, respectively for learning. Figs. 6 and 7 show the training time of the two schemes.

As demonstrated in Figs. 6 and 7, the training time of two schemes is greatly affected by the data size. To be

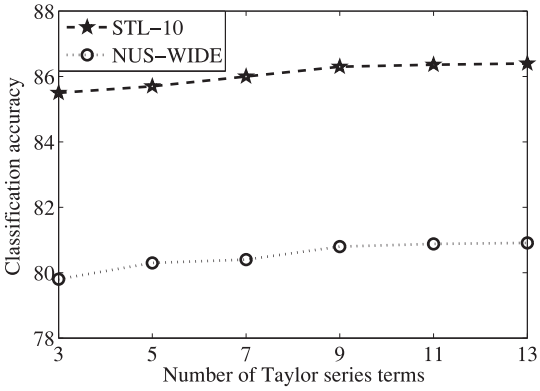


Fig. 5. Classification accuracy with regard to the different number of Taylor series terms.

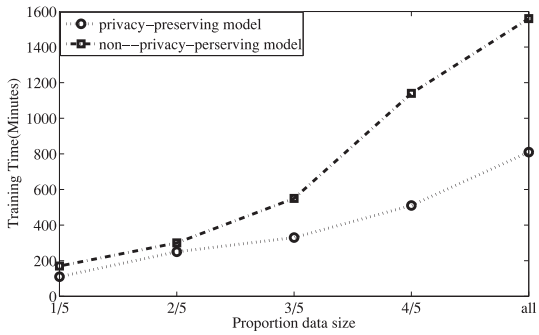


Fig. 6. Training time on the PeMS dataset.

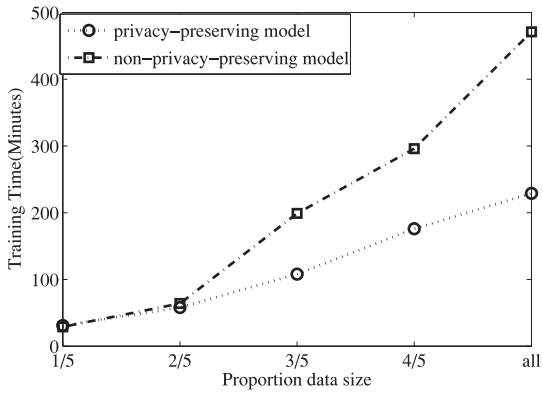


Fig. 7. Training time on the DLeMP dataset.

TABLE 3
Prediction Accuracy of Two Schemes

Periods	C-DCM(%)	PP-DCM(%)
Peak periods	89.3	88.4
Nonpeak period	83.4	81.9

more exact, the training time will increase with the growth of data size. The training time of our proposed scheme includes the data encryption/decryption time, the secure computation time of high-order back-propagation on the cloud and the communication time between the client and the cloud. When the data size is small, our proposed scheme shares almost the same training with the non-privacy-preserving deep computation model. This is because the major time is taken to perform the data encryption/decryption operations and to communicate between the client and the cloud for the small dataset. However, when the proportion of the PeMS data size and the DLeMP data size is larger than 3/5 and 2/5, respectively, the efficiency of our scheme is improved by almost two times than the non-privacy-preserving scheme. This is because the major operation for the large dataset, i.e., the high-order back-propagation, is offloaded to the cloud which can conduct parallel computation for improving the efficiency. In fact, the efficiency of our proposed algorithm can be improved by employing more cloud servers as the data set size increases. So our proposed algorithm is especially suitable for big data feature learning used trained deep computation model.

Then, we compare the proposed scheme to the non-privacy-preserving scheme in terms of prediction accuracy. For the PeMS dataset, three tasks are used here to evaluate the proposed scheme: (1) predicting the traffic flow in peak periods; (2) predicting the traffic flow in nonpeak periods; (3) predicting the traffic flow of peak periods in several time intervals in advance. The results are demonstrated in Table 3 and Fig. 8.

Fig. 9 presents the prediction results of the Dalian gross economic product for 10 years by the two schemes.

As shown in the Table 3, Figs. 8 and 9, our scheme achieves a lower accuracy performance than the non-privacy-preserving deep computation model on the two datasets since the approximation of the activation function introduces the accuracy loss. In particular, our scheme introduces about 1 to 2 percent more error rate compared with the conventional deep computation model. This result

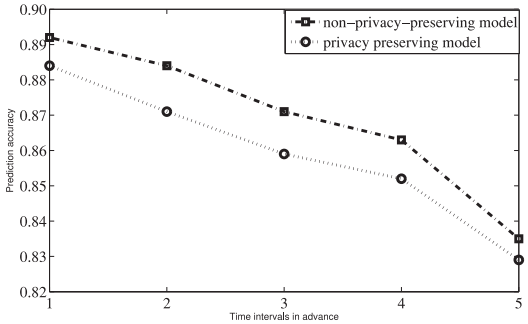


Fig. 8. Prediction accuracy on different time intervals.

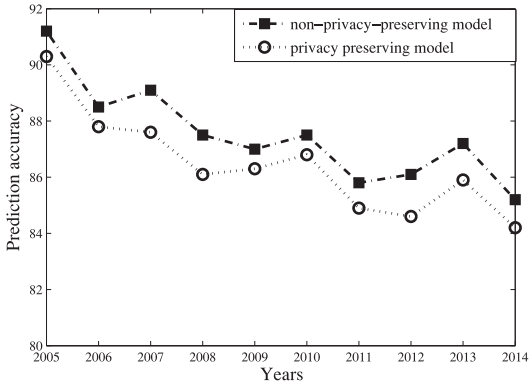


Fig. 9. Prediction accuracy on different years.

is similar to the previous results on the STL-10 dataset and the NUS-WIDE dataset.

Although the accuracy performance of our scheme is a litter lower than that of the non-privacy-preserving deep computation model, it is acceptable in practical use of big data feature learning in the smart city.

Scalability analysis. Finally, we evaluate the scalability of the proposed scheme in terms of the speedup. Speedup is an important criteria to measure the scalability of an algorithm based on cloud computing. Perform the privacy-preserving back-propagation algorithm for learning the parameters of deep computation model on three datasets in the different cloud computing platforms, in which there are 1 node, 5 nodes, 10 nodes, 15 nodes, 20 nodes, respectively. The result is shown as in Fig. 10.

From Fig. 10, the training time of the proposed scheme for training the four three datasets reduces gradually with the increasing number of computing nodes in the cloud computing platform, which demonstrates that adding nodes can significantly improve the system capacity. In particular, for the STL-10 dataset, the PeMS dataset and the DLeMP dataset that are smaller than the NUS-WIDE dataset, the efficiency of our proposed scheme is improved slightly by employing more cloud servers when the number of the cloud nodes is larger than 10. So, 10 cloud nodes are enough for the three training datasets. However, for the NUS-WIDE dataset, the efficiency of our proposed scheme is improved significantly when the number of the cloud nodes is larger than 10. There, our proposed scheme is particularly suitable for big data feature learning since the

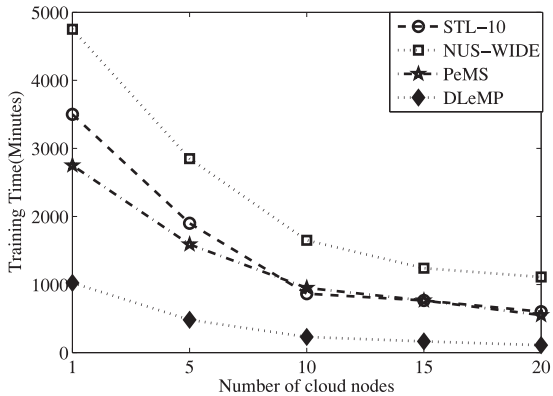


Fig. 10. Training time on different clouds.

performance of our proposed scheme can be further improved by employing more cloud servers for big data.

5 Related Works

In recent years, some privacy preserving methods for data mining and knowledge discovery have been proposed, which can be grouped into two categories: data perturbation method and cryptographic method. The former was proposed by Agrawal and Srikant which preserves the private data by adding noise to the source data [50], while the latter was proposed by Lindell and Pinkas using cryptograhpic methods to prevent the disclosure of the private data [51].

Our proposed scheme belongs to the second group of methods, focusing on privacy preserving deep computation model training and secure big data feature learning by incorporating the strong computing power of the cloud without disclosing the private data. Several works on the problem of privacy preserving neural network learning have been presented recently. For example, Schlitter proposed a privacy preserving back-propagation algorithm for neural network learning that supports horizontal partitioned data [52]. This algorithm only prevent the disclosure of the source data and cannot protect the intermediate results during the learning process. Another scheme proposed by Chen and Zhong protects simultaneously the source data and the intermediate results [53]. Moreover, the solution supports vertically partitioned data while the scheme proposed by Schlitter supports horizontal partitioned data. Bansal et al. proposed a privacy preserving back-propagation neural network learning algorithm for arbitrarily partitioned data [54]. The above schemes just support the two-party scenario. They cannot be applied to the multiparty setting because directly extending them to the multiparty setting will increase a large of communication overhead. To overcome this limitation, Yuan and Yu introduced a privacy preserving back-propagation neural network learning algorithm for multiparty scenario which keeps the computation and communication costs on each party minimal and independent to the number of participants by using the power of cloud computing [21]. Another advantage of this scheme is that it supports arbitrarily partitioned data like the scheme proposed by Bansal et al. Other methods focusing on privacy preserving back-propagation algorithm can be found in [55] and [56].

There are at least two notable differences between our work and the above schemes. The current schemes have proposed for the privacy preservation of the conventional back-propagation neural network learning while our method focuses on the secure high-order back-propagation algorithm for deep computation model learning proposed in our previous work. Furthermore, the above schemes are designed for secure multiparty collaboration which conducts joint back-propagation neural network learning on the union of their respective datasets. Different from them, the goal of our scheme is to improve the efficiency of deep computation model learning by incorporating the computing power of the cloud without disclosing the private data. Our scheme is especially suitable for big data learning since the performance can be improved further by employing more cloud servers when the data size grows.

6 Conclusion

Big data offers the great opportunities and transformative potential for various areas such as e-commerce, healthcare industry manufacturing, social network and educational services. Therefore, deep computation, a novel area, has attracted great interests of researchers in recent years. It refers to a systematical model for big data representation, storage, analytic and mining based on tensor theory. In this paper, we proposed a privacy preserving deep learning model for big data feature learning by incorporating the computing power of the cloud. The proposed scheme uses the BGV encryption scheme to support the secure computation operations of the high-order back-propagation algorithm efficiently for deep computation model training on the cloud. In our scheme, only the encryption operations and the decryption operations are performed by the client while all the computation tasks are performed on the cloud.

Experimental results clearly demonstrated that: (1) our scheme can efficiently deal with deep computation model for big data feature learning by incorporating the high computing power of the cloud without disclosing private data; (2) although our scheme takes additional overhead to perform the data encryption/decryption and communication between the client and the cloud, it is still more efficient than the conventional deep computation model. More importantly, the performance of our scheme can be further improved by employing more cloud servers, which is particularly suitable for big data, thanks to the high scalability of the cloud; (3) although the accuracy performance of our scheme is a litter lower than that of the non-privacy-preserving deep computation model, it is acceptable in practical use of big data feature learning in smart city. Future work focuses on the design of the incremental deep computation model to improve the efficiency of big data computing in the smart city.

References

[1] W. Rong, X. Zhang, C. Dave, L. Chao, andS. Hao, "Smart city architecture: A technology guide for implementation and design challenges," *China Commun.*, vol. 11, no. 3, pp. 56–69, 2014.

[2] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 112–121, Apr. 2014.

[3] A. Domingo, B. Bellalta, M. Palacin, M. Oliver, and E. Almirall, "Public open sensor data: Revolutionizing smart cities," *IEEE Technol. Soc. Mag.*, vol. 32, no. 4, pp. 50–56, Dec. 2013.

[4] G. Pan, G. Qi, W. Zhang, S. Li, Z. Wu, and L. T. Yang, "Trace analysis and mining for smart cities: Issues, methods, and applications," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 120–126, Jun. 2013.

[5] I. Vilajosana, J. Llosa, B. Martinez, M. Domingo-Prieto, A. Angles, and X. Vilajosana, "Bootstrapping smart cities through a self-sustainable model based on big data flows," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 128–134, Jun. 2013.

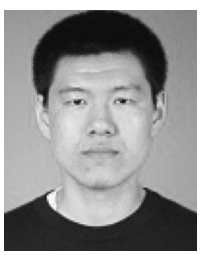
[6] X. -W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, May 2014.

[7] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics: (Statistical) learning tools for our era of data deluge," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 18–31, Sep. 2014.

[8] T. Condie, P. Mineiro, N. Polyzotis, and M. Weimer, "Machine learning on big data," in *Proc. IEEE Int. Conf. Data Eng.*, 2013, pp. 1242–1244.

[9] X. Wu, X. Zhu, G. -Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[10] H. H. Huang and H. Liu, "Big data machine learning and graph analytics: Current state and future challenges," in *Proc. IEEE Int. Conf. Big Data*, 2014, pp. 16–17.
[11] C. Yang, C. Liu, X. Zhang, S. Nepal, and J. Chen, "A time efficient approach for detecting errors in big sensor data on cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 329–339, Feb. 2015.
[12] X. Zhu, X. Qin, and M. Qiu, "QoS-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters," *IEEE Trans. Comput.*, vol. 60, no. 6, pp. 800–812, Jun. 2011.
[13] A. S. Prasad and S. Rao, "A mechanism design approach to resource procurement in cloud computing," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 17–30, Jan. 2014.
[14] S. M. Kwang, "Agent-based cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 564–577, Oct.-Dec. 2012.
[15] N. Jones, "Computer science: The learning machines," *Nature*, vol. 505, no. 7482, pp. 146–148, 2014.
[16] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, Jun. 2011.
[17] X. Zhu, C. Chen, Laurence T. Yang, and Y. Xiang, "ANGEL: Agent-based scheduling for real-time tasks in virtualized clouds," *IEEE Trans. Comput.*, [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7054494, 2015.
[18] Q. Zhang and Z. Chen, "A weighted kernel possibilistic C-means algorithm based on cloud computing for clustering big data," *Int. J. Commun. Syst.*, vol. 27, no. 9, pp. 1378–1391, 2014.
[19] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *Proc. VLDB Endowment*, vol. 3, no. 1–2, pp. 703–710, 2010.
[20] J. Lin and A. Kolcz, "Large-scale machine learning at Twitter," in *Proc. ACM Conf. Manage. Data*, 2012, pp. 793–804.
[21] J. Yuan and S. Yu, "Privacy preserving back-propagation neural network learning made practical with cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 212–221, Jan. 2014.
[22] J. Bacon, D. Eyers, T. F. J. -M. Pasquier, J. Singh, J. I. Papagiannis, and P. Pietzuch, "Information flow control for secure cloud computing," *IEEE Trans. Netw. Serv. Manage.*, vol. 11, no. 1, pp. 76–89, Apr. 2014.
[23] E. Z. Tragos, V. Angelakis, A. Fragkiadakis, D. Gundlegard, C.-S. Nechifor, G. Oikonomou, H. C. Pohls, and A. Gavras, "Enabling reliable and secure IoT-based smart city applications," in *Proc. Int. Conf. Pervasive Comput. Commun.*, 2014, pp. 111–116.
[24] (2013). The health insurance portability and accountability act of privacy and security rules [Online]. Available: http://www.hhs.gov/ocr/privacy
[25] Y. Lindelland B. Pinkas, "Privacy preserving data mining," *J. Cryptol.*, vol. 15, no. 3, pp. 177–206, 2002.
[26] W. Du and Z. Zhan, "Using randomized response techniques for privacy preserving data mining," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 505–510.
[27] R. Wright and Z. Yang, "Privacy-preserving Bayesian network structure computation on distributed heterogeneous data," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 713–718.
[28] J. Vaidya and C. Clifton, "Privacy preserving naive Bayes classifier for vertically partitioned data," in *Proc. SIAM Int. Conf. Data Mining*, 2004, pp. 522–526.
[29] S. Laur, H. Lipmaa, and T. Mielikaihen, "Cryptographically private support vector machines," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 618–624.
[30] J. Vaidya and C. Clifton, "Privacy-preserving K-means clustering over vertically partitioned data," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 206–215.
[31] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. ACM Innovations Theoretical Comput. Sci. Conf.*, 2012 pp. 309–325.
[32] L. Kuang, F. Hao, L. T. Yang, M. Lin, C. Luo, and G. Min, "A tensor-based approach for big data representation and dimensionality reduction," *IEEE Trans. Emerging Topics Comput.*, vol. 2, no. 3, pp. 280–291, Sep. 2014.
[33] A. Cichocki, "Era of big data processing: A new approach via tensor networks and tensor decompositions," *arXiv preprint arXiv*:1403.2048, 2014.
[34] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[35] Y. Liu, Y. Liu, and K. Chan, "Tensor distance based multilinear locality-preserved maximum information embedding," *IEEE Trans. Neural Netw.*, vol. 21, no. 11, pp. 1848–1854, Nov. 2010.
[36] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978.
[37] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[38] D. Boneh, E. -J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proc. Int. Conf. Theory Cryptography*, 2005, pp. 325–341.
[39] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. ACM Symp. Theory Comput.*, 2009, pp. 169–178.
[40] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security*, 1998, pp. 308–318.
[41] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.
[42] I. Damgard and M. Jurik, "A length-flexible threshold cryptosystem with applications," in *Proc. Australasian Conf. Inf. Security Privacy*, 2001, pp. 350–364.
[43] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1985, pp. 10–18.
[44] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Proc. 32nd Annu. Cryptol. Conf. Adv. Cryptol.*, 2012, pp. 868–886.
[45] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proc. 33rd Annu. Cryptol. Conf. Adv. Cryptol.*, 2013, pp. 75–92.
[46] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.
[47] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUSWIDE: A real-world web image database from National University of Singapore," in *Proc. ACM Int. Conf. Image Video Retrieval*, 2009, pp. 1–9.
[48] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct. 2014.
[49] (2015). Performance Measurement System (PeMS) [Online]. Available: http://pems.dot.ca.gov
[50] D. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proc. ACM Conf. Manage. Data*, 2000, pp. 439–450.
[51] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *Lecture Notes Comput. Sci.*, vol. 1880, pp. 36–45, 2000.
[52] N. Schlitter, "A protocol for privacy preserving neural network learning on horizontal partitioned data," in *Proc. Privacy Statist. Databases*, 2008, pp. 298–315.
[53] T. Chen and S. Zhong, "Privacy-preserving backpropagation neural network learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 10, pp. 1554–1564, Oct. 2009.
[54] A. Bansal, T. Chen, and S. Zhong, "Privacy preserving back-propagation neural network learning over arbitrarily partitioned data," *Neural Comput. Appl.*, vol. 20, no. 1, pp. 143–150, Feb. 2011.
[55] M. Dong, H. Li, K. Ota, and H. Zhu, "HVSTO: Efficient privacy preserving hybrid storage in cloud data center," in *Proc. IEEE Conf. Comput. Commun. INFOCOM Workshop Security Privacy Big Data*, 2014, pp. 529–534.
[56] M. Barni, C. Orlandi, and A. Piva, "A privacy-preserving protocol for neural-network-based computation," in *Proc. 8th Workshop Multimedia Security*, 2006, pp. 146–151.



Qingchen Zhang received the bachelor's and master's degrees in Southwest University, China. He is currently working toward the PhD degree at the School of Software Technology in the Dalian University of Technology (DLUT), China. His research interests include big data and deep learning.

Laurence T. Yang is a professor at the School of Computer Science in the Huazhong University of Science and Technology, China and at the Department of Computer Science in St. Francis Xavier University, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous computing, and big data.

Zhikui Chen is a professor at the School of Software Technology in the Dalian University of Technology, China. He is leading the Institute of Ubiquitous Networks and Computing of the Dalian University of Technology. His research area includes internet of things and big data processing.