IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 2 September 2022, accepted 20 September 2022, date of publication 26 September 2022, date of current version 13 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3210134

RESEARCH ARTICLE

CHE: Channel-Wise Homomorphic Encryption for Ciphertext Inference in Convolutional Neural Network

TIANYING XIE, (Student Member, IEEE), HAYATO YAMANA, (Member, IEEE), AND TATSUYA MORI, (Member, IEEE)
Department of Computer Science and Communications Engineering, Waseda University, Tokyo 169-8555, Japan

Corresponding author: Tatsuya Mori (mori@nsl.cs.waseda.ac.jp)

This work was supported in part by the China Scholarship Council (CSC) under Grant 202108050217, and in part by the Japan Science and Technology Agency (JST) Core Research for Evolutional Science and Technology (CREST) under Grant JPMJCR1503.

ABSTRACT Privacy-preserving deep learning (PPDL), which leverages Homomorphic Encryption (HE), has attracted attention as a promising approach to ensure the privacy of deep learning applications' data. While recent studies have developed and evaluated the HE-based PPDL algorithms, the achieved performances, such as accuracy and latency, need improvement to make the applications practical. This work aims to improve the performance of the image classification of HE-based PPDL by combining two approaches — Channel-wise Homomorphic Encryption (CHE) and Batch Normalization (BN) with coefficient merging. Although these are commonly used schemes, their detailed algorithms and formulations have not been clearly described. The main contribution of the current study is to provide complete and reproducible descriptions of these schemes. We evaluate our CHE and BN implementation by targeting the Cheon-Kim-Kim-Song scheme as an HE scheme and Convolution Neural Network (CNN) as a machine learning scheme while using the MNIST and CIFAR-10 as the datasets. In addition, we compare the results with the five state-of-the-art neural network architectures. Our experiments demonstrate that the CHE can serve as a tool for empirically achieving shorter latency (the shortest 7.76 seconds) and higher accuracy (the highest 99.32%) compared with the previous studies that aimed to establish the classification of the encrypted MNIST data with CNN. Our approach can aid in designing a more robust and flexible PPDL.

INDEX TERMS Channel-wise, homomorphic encryption, CNN, privacy-preserving, ciphertext inference.

I. INTRODUCTION

Users upload data to cloud servers to make use of a service such as a training machine learning model. These are known as Machine Learning as a Service (MLaaS). However, the users can be reluctant to upload their personal and sensitive data as the server might not be secure. Presently, servers leverage encryption to protect the privacy of user data. Homomorphic Encryption (HE), which can perform arithmetic computations on ciphertext, is one of the promising Privacy-preserving Deep Learning (PPDL) methods.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Huang.

Although there have been several studies [1], [2] [3], [4] [5], [6], [7] on HE-based PPDL, the achieved latency was not short enough, and the accuracy was not high enough. Gilad *et al.* [1] developed ciphertext inference of the MNIST dataset, but the accuracy was 96%, and the latency was 570 seconds. Furthermore, Ishiyama *et al.* [4] achieved an accuracy of 99.18% and a latency of 21.15 seconds on the MNIST. We aim to improve the performance by targeting image classification of HE-based PPDL by combining two approaches: Channel-wise Homomorphic Encryption (CHE), and Batch Normalization (BN), with coefficient merging (CM).

Most previous studies [1], [4], [6] have adopted Pixel-wise Homomorphic Encryption (PiHE) for image classification, where multiple images are packed together and processed at once. However, we argue that this approach may not be suitable for all practical applications as the user may want the MLaaS to process only one or a few images, not a batch of images. To implement empirical application and improve processing efficiency, we aim to leverage the CHE, which provides shorter inference latency because it processes the data in channel instead of in pixel. It packs elements of one channel into a single ciphertext forming a vector, which can perform the Single Instruction Multiple Data (SIMD) operations. Aharoni *et al.* [8] introduced a new packing-oblivious programming framework, which somewhat generalized the CHE and the PiHE. Although Dathathri *et al.* [9] and Lou *et al.* [10] have previously proposed a conceptually similar CHE; however, the algorithm was not described in detail. Given this background, we formulate the algorithm and computation scheme ‘‘Onion’’ to achieve ciphertext inference in Convolutional Neural Network (CNN) through the CHE approach. The BN layer is widely used in the neural network (NN) model to achieve high accuracy [11], whitening the input images via the shift and the bias. Chabanne *et al.* [2] placed the BN layer before the activation layer to attain a restricted stable distribution at the entry of the Rectified Liner Activation (ReLU). HEMET [10] set the BN layer behind the activation layer and obtained excellent performance. Moreover, Ishiyama *et al.* [4] placed the BN layer in front of the activation layer. However, there is still not sufficient evidence to decide if it is better to place the activation layer before the BN layer or vice versa.

We have pointed out the two schemes with different orders of the convolution layer, activation layer, and BN layer, namely, CBA (Convolution-BN-Activation) and CAB (Convolution-Activation-BN). Each scheme is a method of CM, and mathematical formulae are given to demonstrate the scheme's stability. The latency is related to the number of predefined levels, so naively deploying the BN layer increases the multiplicative depth (MD); since the number of multiplications and rotations are highly related to the execution time, which exploits a higher level, it leads to longer latency. We have fused the BN and activation layer with the CM as a ‘‘Mapping’’ layer, which increases accuracy and decreases the latency of ciphertext inference, and describes the functions of the two schemes. In this article, we leverage the [.] symbol for ciphertext to distinguish more clearly between plaintext and ciphertext.

Our contributions are summarized as follows:

- We propose the explicit algorithms of the CHE to address the HE-based ciphertext inference problem in the CNN for higher accuracy and shorter latency.
- We formulate the BN layer with CM in two schemes (CAB and CBA) and show the derivation of mathematical formulae, and the two schemes can reduce ciphertext inference latency without increasing the MD.
- We evaluate and compare the proposed CHE against five works [1], [2], [3], [4], [6], and achieve 99.30% of accuracy and 7.76 seconds of latency on MNIST, 76.40% of accuracy and 111.91 seconds of latency on CIFAR-10, the result verifies that the CHE is practical to the end-user during MLaaS.

The remainder of the paper is structured as follows. Section II provides the related work, the related research background for PPDL and HE, and the threat model. Then, Section III contains the proposed CHE from two aspects: algorithms of the CNN layers and CM schemes. Our experimental results are described in Section IV. The explanations, limitations, and significance of the work are discussed in Section V. Section VI concludes the paper.

II. BACKGROUND

A. RELATED WORK

The research in the PPDL area can be divided into five camps that implement different X-based methods: HE-based, Secure Multi-Party Computation (MPC)-based, Differential Privacy (DP)-based, Secure Enclaves (SE)-based, and Hybrid-based.

CryptoNets [1] is the first work to use the HE-based method to make ciphertext inference in the NN model. As CryptoNets became ineffective for deeper NN, Chabanne *et al.* [2] designed and evaluated the first privacy-preserving classification in NN. To protect privacy without accuracy loss, Juvekar *et al.* [12] and Zhang *et al.* [13] have proposed using the interactive paradigm to split the NN model into two parts. Furthermore, Hesamiford *et al.* [3] deployed the NN model on the server in the non-interactive paradigm, placing the activation function with the polynomials. Due to time-consuming PPDL inference, Lou *et al.* [10] proposed implementing mobile NN models to achieve shorter inference latency and higher inference accuracy. In TenSEAL [5], the authors implement the image to column (im2col) method to process the convolution layer. However, it is no way to find simple way to build a network with more than one convolution layer, which is the non-interactive scheme.

The Secure MPC-based method exploited Yao's garbled circuits for secure two-party computation. Mohassel *et al.* [14] developed a combination of the garbled circuit with oblivious transfer and secret sharing. Rizai *et al.* [15] suggested a distributed training computation with a novel approach to secret sharing. Furthermore, Liu *et al.* [16] devised a distributed Secure MPC framework for privacy-preserving data mining, with one-hot encoding and a lower-upper decomposition algorithm.

The DP-based method adds random noise to a dataset and generates fake training data. PATE [17] originated a DP learning process utilizing teacher and student models. Fan *et al.* [18] initiated a local DP framework for data centers using the Laplacian mechanism to measure the quality of privacy protection.

The Secure Enclaves-based PPDL is the method by which a client sends data to the secure enclave environment where all of the data and models are hidden from the outside world, such as [19]. The Hybrid-based PPDL is the method that leverages several techniques to build a secure and privacy-preserving model, such as GAZELLE [12].

Kim *et al.* [20] reported a comprehensive survey about PPDL, showing a high-level view of the PPDL research.

B. HOMOMORPHIC ENCRYPTION

HE is an asymmetric cryptographic method that facilitates arithmetic computations over encrypted data (ciphertext). Since the HE operations introduce a certain amount of noise into the ciphertext, when the accumulated noise grows beyond a noise budget, HE decryption does not resolve the correct result [5]. Although an operation named bootstrapping, proposed by Gentry [21], could reduce the accumulated noise in a ciphertext, it is too time-consuming for practical applications. To address this limitation, Cheon *et al.* [22] proposed the approximate HE with the Residue Number System (RNS) and named the RNS-variant as Cheon-Kim-Kim-Song (RNS-CKKS) HE scheme. It is a leveled HE scheme, setting a threshold for noise budget to compute finite HE operations without bootstrapping. The RNS-CKKS HE scheme denotes the degree of polynomial modulus by N, which is a power of 2, encodes data to $N/2$ fixed-point numbers (if not enough, pad with 0), and then encrypts $N/2$ numbers into $N/2$ slots of one ciphertext.

The followings are the core algorithm of the RNS-CKKS HE scheme: 1) Encode operation encodes the input vector into the polynomials for encryption, 2)HE operations include HE addition, HE multiplication, and HE rotation: a)HE addition makes element-wise addition for slots in the corresponding position; b)HE multiplication performs element-wise multiplication; c)HE rotation rotates the slots of ciphertext, similar to the left and right panning. A HE operation is performed simultaneously on all slots of the ciphertext. A ciphertext is a polynomial of degree N with each coefficient representing modulo Q, where Q is a product of n primes, describing the level of the HE. To reduce the noise, a rescaling operation is needed to convert n-level ciphertext to $(n-1)$-level ciphertext. The representative encode function and rotation function are shown as follows:

- $Encode(z) \longrightarrow p$: CKKS exploits the rich structure of integer polynomial rings for its plaintext and ciphertext spaces. Nonetheless, data comes more often in the form of vectors than in polynomials. It is necessary to encode input vector z into a polynomial p.
- $Rot([c], n) \longrightarrow [c']$: Deploy linear rotation over ciphertext $[c]$, n is the step size, which cycles ciphertext horizontally to the left. n starts from 1 rather than 0, and this function needs one key, "Galois key," set as default.

C. PRIVACY-PRESERVING DEEP LEARNING

Security and privacy preservation of the data must be considered to convince the user to upload their data to a server. HE allows data to be processed in ciphertext, maintaining a trade-off between privacy and performance. To protect the user's privacy, the privacy-preserving NN is built to perform inference directly on encrypted data; the ciphertext inference can be made over two paradigms: interactive paradigm (online mode), such as GELU-NET [13] (the model was partitioned NN into two non-colluding parties), GAZELLE [12] (high accuracy but huge memory cost), and BAYHENN [23] (the inference was partitioned into linear and non-linear computations), and non-interactive paradigm (offline mode), such as CryptoNets [1] (the first work developed in this area), Chabanne *et al.* [2] (first proposed work that combines activation function with BN function), Hesamiford *et al.* [3] (suggested using derivative of polynomials to replace the activation function), Ishiyama *et al.* [4] (proposed using CM for decreasing MD), Xie *et al.* [24] (exploited the Efficient Integer Vector HE in CNN) and Dathathri *et al.* [9] (initiated an optimizing compiler for HE NN inference). In this work, the focus is on ciphertext inference in the non-interactive paradigm. The Table. 1 shows the differentiation and comparison of PiHE and CHE. Although the CHE and the PiHE obtain similar accuracy, the CHE can shorten the latency and reduce the consumed memory, but it will smaller the throughput.

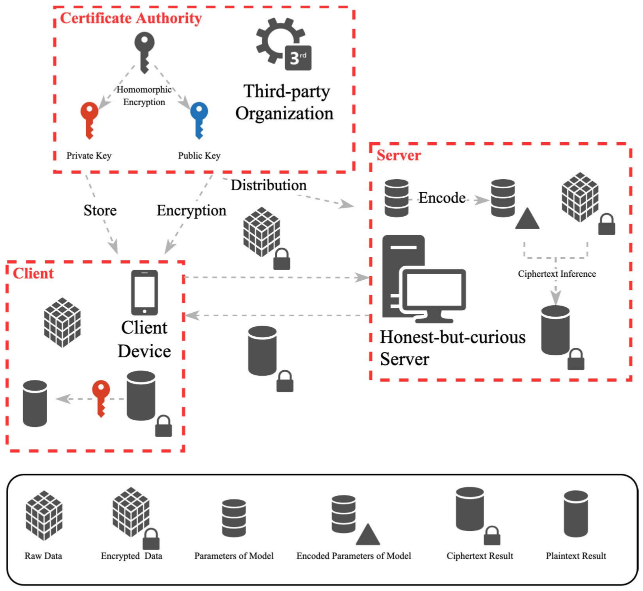


FIGURE 1. Threat model. There is no collusion. The server is honest-but-curious; it can access the plaintext machine learning model and perform ciphertext inference.

D. THREAT MODEL

The threat model is referenced from prior PPDLs [9], [10]. We assume that the machine learning server is a potential attacker, which is honest-but-curious, implying that the server does not deviate from the defined protocol but attempts to learn all the possible information from legitimately received messages [26]. To be more realistic and safe, instead of generating the public key and private key by the user, we assume that keys are distributed by a trusted third-party organization, which is different from previous threat models. The third-party certificate authority generates and distributes the user's public key and private key. The data sent to the server

TABLE 1. Differentiation and comparison of PiHE and CHE. Note that we assume pixels and channels are less than or equal to the slot size.

Encryption Method	Related Work	Accuracy	Latency	Throughput	Memory Requirement	Number of Ciphertext
Pixel-wise HE	[1] [2] [3] [4] [6] [13] [23] [24] [25] [15] [16] [17]	Same level	Long	High	Larger	Number of pixels in an image
Channel-wise HE	[5] [7] [9] [10] [12] CHE		Short	Small	Smaller	Number of channels

is encrypted by the HE public key and privacy-preserved by the encryption method. The model is trained using plaintext data, and parameters are stored in the server without revealing them to the user. The server accesses the plaintext parameters, which causes data leakage. It performs private and secure inference over encrypted data without decryption or accessing the secret key. Only the client decrypts the ciphertext result by the local stored secret key, as shown in Fig. 1.

III. METHODOLOGY

We propose a detailed and explicit CHE to run inference over encrypted data in CNN and its workflow (Section III-A). Moreover, we provide a high-level description of the BN layer with CM (Section III-B).

A. ALGORITHMS OF CHE

In the proposed method, the image data is preprocessed channel-wise instead of pixel-wise. In practice, the preprocessing depends on the inference. Technically, the PiHE processes data in a matrix (tensor), while the CHE exploits a vector. We introduce the data preprocessing scheme (Section III-A1) and functions of CNN layers (Sections III-A2 to III-A7) by deploying the CHE. The details of different layers of methods over vector and tensor are shown in Table 2. The entire data processing schemes of the CNN model are shown in the Fig. 2. It will be more comprehensible and readable to understand the following detailed designs and processes of the model, which uses the proposed CHE.

1) PRE-PROCESS AND ENCRYPT DATA

The image data consists of tensors from three channels. In the PiHE, the same pixels of multiple images are packed in a single ciphertext, and the number of ciphertexts equals the number of images. In contrast, in the CHE, the first step is to flatten each channel of an image into a vector, then pack and encrypt each vector into one ciphertext, which is fed as the input to the NN model. In this case, the number of ciphertexts equals the number of channels of the image. For a single image, the CHE takes less time to transform the image into a ciphertext.

2) RESHAPE CONVOLUTION LAYER

To deploy a deeper NN model with multi-convolution layers in a non-interactive scheme, we split the convolution layer into two parts: *rotation & accumulation*, and *mask computation*. Suppose the input has C channels, a height of H, and a width of W. It can then be encrypted into C ciphertexts (assuming that the product of H and W is less than or equal to the slot size $\frac{N}{2}$), which packs $H \times W$ input elements (pixels).

In *rotation & accumulation*, the HE rotation operation $Rot([c], n)$ is required to compute element-wise HE multiplication between the input and weight filter, where $[c]$ is ciphertext and n represents the stride for rotation. Obtaining corresponding n values is an essential step of the convolution layer, which has not been described in previous works [9], [10]. We propose the algorithm-Rotation Index Searching (RIS) to find n values, as depicted in Algorithm 1. Using a list of n values by the HE rotation operation $Rot([c], n)$, the model can calculate element-wise HE multiplication between the input and each weight filter and accumulate the results into a single ciphertext. The result of a single ciphertext is a semi-manufactured output with two weaknesses: irrelevant slots in ciphertext, called noise, and chaotic structure of ciphertext compared with the input. In *mask computation*, a plaintext mask replaces the values of valid and irrelevant slots with 1 and 0, respectively, to remove noise. Note that the mask variable is a temporary vector with 0s and 1s, where 0s represent redundant slots, and 1s represent valid slots.

Next, the indexes of valid slots must be calculated to rearrange the chaotic structure. For this, we propose the algorithm-Valid Index Searching (VIS) that is used to prepare the mask, recorded in Algorithm 2, and the return of the algorithm represents the indexes of 1s.

After implementing *mask computation*, a well-structured vector (with valid values listed one after another) is necessary to maintain the exact structure of the input and output. We propose the algorithm-Chaotic Rotation Index (CRIS) to look for valid values, as shown in Algorithm 3. However, *mask computation* is time-consuming and unsuitable for speedup inference. The method mentioned above of processing a chaotic vector for the next layer may not be necessary for the convolution layer that we argue the model does not need to get the well-structured output. Instead, we propose a computation scheme, "Onion," shown in Fig. 4, which relates to Algorithm 1 and Algorithm 2. The computation scheme "Onion" produces a chaotic vector as the output (a not well-structured vector) of the convolution layer. The *rotation & accumulation* (calculate with filter) and *mask computation* (remove noise) cost the same as the one MD in ciphertext; thus, the MD of the convolution layer is two.

The description follows from Fig. 3. Consider a 4×4 orange-colored matrix A filled with numbers from 1 to 16 left to right and top to bottom. In the first 2D convolution layer, a single 2×2 filter F with the stride $s(1, 1)$ samples the upper left corner element. After convolution, the ideal output is presented as numbers in a blue-colored solid matrix B, and the practical output is in a dotted matrix. Using the algorithm VIS, the model can get a list l_1 of indexes of valid values.

TABLE 2. Different functions of vector and tensor. Tensors are employed for pixel-wise approach and vectors are employed for channel-wise approach.

Function	Data	
	Vector	Tensor
Convolution	Weighted sum between input vector(s) and filter(s) with HE rotation	Calculate a cross-correlation between an input and a filter tensor
Element-wise operation	The same as "tensor," also includes activation with approximated polynomials	Such as ReLU, Batch Normalization, etc.
Pooling	Data is serial data, which has to simulate combining adjacent elements of tensor; also is the weighted sum between input tensor(s) and filter(s) without various weights, instead, fixed weights whose values equal to $\frac{1}{f\times f}$; almost the same as "convolution"	Combine adjacent elements using a reduction operation such as maximum or average; also called down-sampling
Reshaping	Pack all outputs (ciphertexts) of the last layer into one ciphertext preceding the Fully Connected layer	Reinterpret the shape of a tensor, for example, to flatten preceding a matrix multiplication
Matrix multiplication	Since ciphertext is a vector, there is no way for matrix multiplication. Instead, it is vector-matrix multiplication that calculates vector (ciphertext) and filter (plaintext matrix)	Represent neurons that are connected to all inputs, which are found in a dense layer typically at the end of a CNN

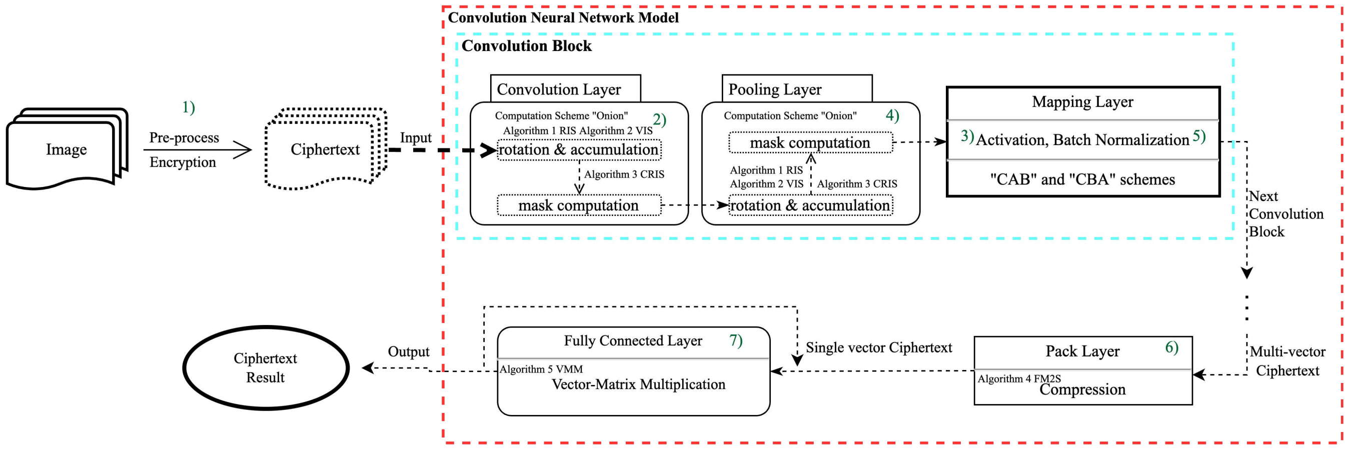


FIGURE 2. The flow chart showing the entire data processing schemes of the CNN model.

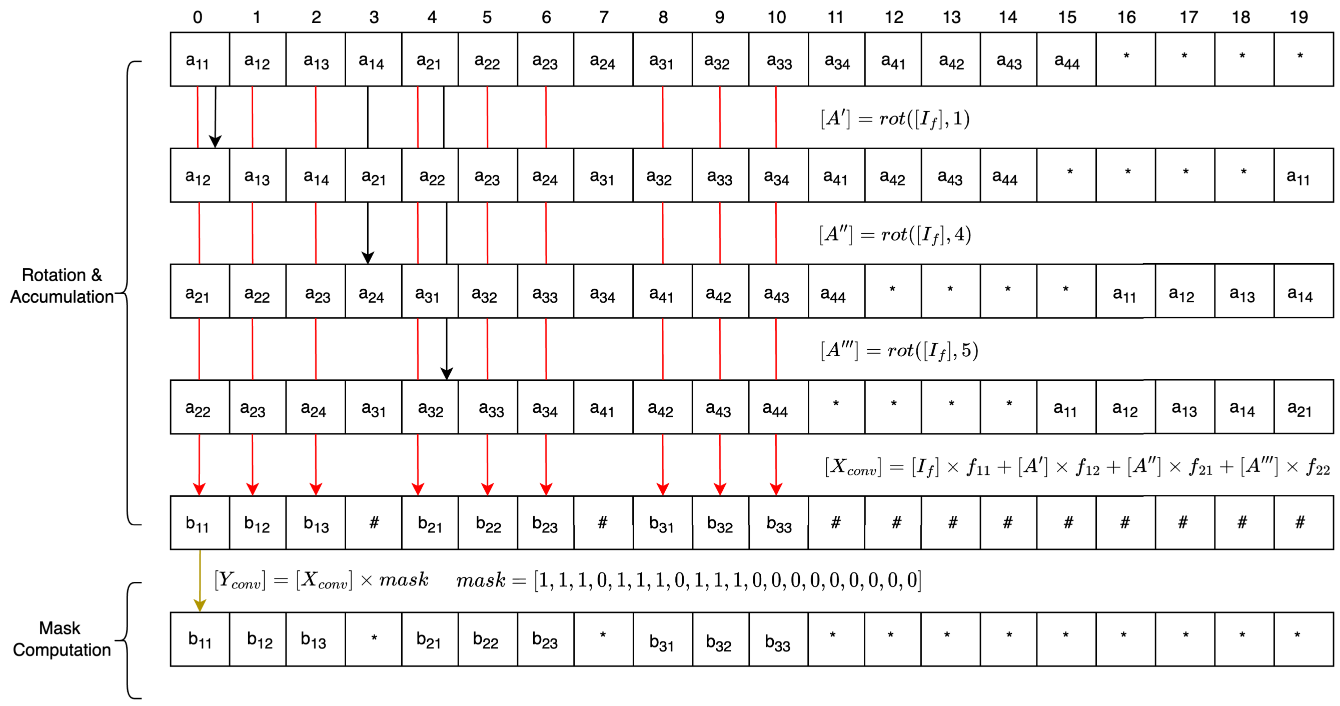

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
a11 a12 a13 a14 a21 a22 a23 a24 a31 a32 a33 a34 a41 a42 a43 a44 * * * *
[A'] = rot([I_f], 1)
a12 a13 a14 a21 a22 a23 a24 a31 a32 a33 a34 a41 a42 a43 a44 * * * * a11
[A''] = rot([I_f], 4)
a21 a22 a23 a24 a31 a32 a33 a34 a41 a42 a43 a44 * * * * a11 a12 a13 a14
[A'''] = rot([I_f], 5)
a22 a23 a24 a31 a32 a33 a34 a41 a42 a43 a44 * * * * a11 a12 a13 a14 a21
[X_conv] = [I_f] × f11 + [A'] × f12 + [A''] × f21 + [A'''] × f22
b11 b12 b13 # b21 b22 b23 # b31 b32 b33 # # # # # # # # #
[Y_conv] = [X_conv] × mask   mask = [1,1,1,0,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0]
b11 b12 b13 * b21 b22 b23 * b31 b32 b33 * * * * * * * * *
Rotation & Accumulation
Mask Computation

FIGURE 3. Convolution Layer over Ciphertext. Assume total slots are 20, the input size is 4 × 4, and the filter size is 2 × 2 with stride 1. "#" means the redundant slot, and "*" means the slot holding zero or relatively small noise.

Similarly, the second layer is sampled by filter F. The output of this layer is presented in a purple-colored matrix C, and the index list l_2 is calculated by Algorithm 2. The objective of the model is to obtain the indexes of all the valid values.

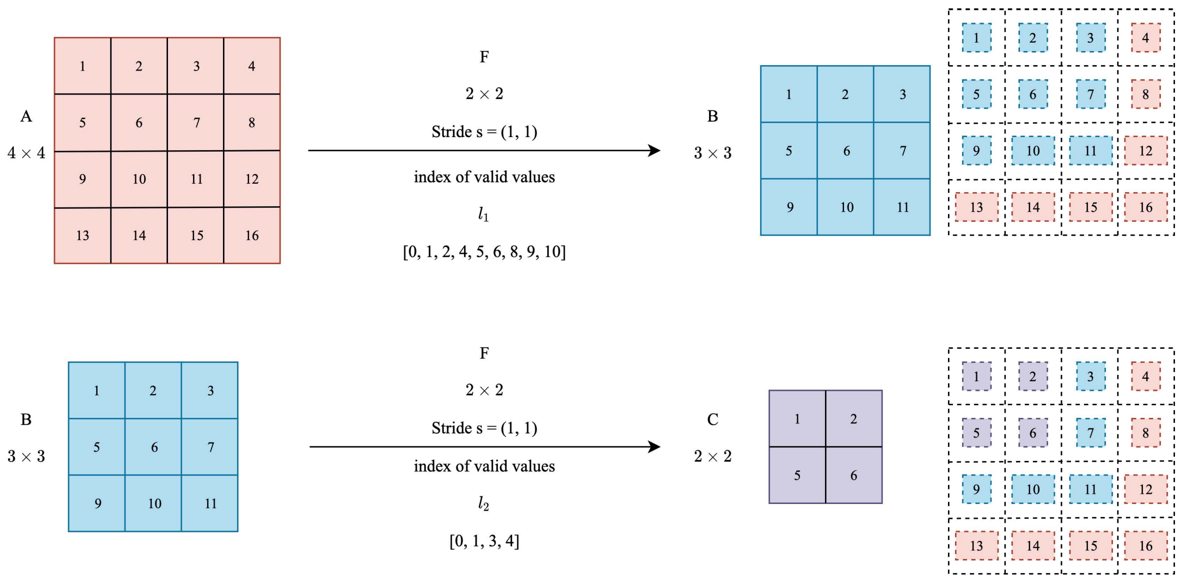


FIGURE 4. "Onion" computation scheme of CHE shows how to calculate the ciphertext in the model without revealing the well-structured vector.

However, elements in l_2 cannot be used as subscripts of the output vector to find valid values. Instead, the elements in l_2 are regarded as subscripts in l_1, and the elements corresponding to these subscripts in l_1 then form the index set. The model leverages elements in the index set as the subscripts of the output vector. Hence, the model can find the required indexes, layer by layer, like an onion. Notably, the model must store an index list, passing and updating layer by layer, during the inference.

Algorithm 1 Rotation Index Searching (RIS)

Input: input feature map size m, convolution filter size f

Output: indexing list ris_idx

```
1: for $i = 0, \ldots, f-1$ do
2:     Set line anchor $start \longleftarrow i \times m$
3:     for $j = 0, \ldots, f-1$ do
4:         If $i = f-1 \& j = f-1$ : $continue$
5:         Compute first point of line $fp$: $fp \leftarrow start + j$
6:         If $j == f-1$:
7:             $ris\_idx_i \leftarrow (i+1) \times m$
8:         Else: $ris\_idx_i \leftarrow fp + 1$
9:     end for
10: end for
11: return $ris\_idx$
```

3) ACTIVATION LAYER

HE data only supports the polynomial function. Thus, the activation function should be approximated by a polynomial function to obtain high accuracy. The approximation method depends on the desired performance of the model, and there have been several studies [4], [25], [27]. As the method of approximating the polynomial is not the objective of the current study, the square function is used to replace the ReLU. This square function is a second order polynomial of the form $ax^2 + bx + c$, where $a = 1, b = 0, c = 0$. The MD of the activation layer is 1.

Algorithm 2 Valid Index Searching (VIS)

Input: input feature map size m, output feature map size n, layer's stride s

Output: indexing list vis_idx

```
1: for $i = 0, \ldots, n-1$ do
2:     for $j = 0, \ldots, n-1$ do
3:         $vis\_idx_i \leftarrow i \times s \times m + j \times s$
4:     end for
5: end for
6: return $vis\_idx$
```

Algorithm 3 Chaotic Rotation Index Searching (CRIS)

Input: input feature map size m, output feature map size n, layer's stride s

Output: indexing list rot_idx

```
1: Indexing list $idx = VIS(m, n, s)$
2: for $i = 1, \ldots, n^2 - 1$ do
3:     $rot\_idx_i \leftarrow idx_i - i$
4: end for
5: return $rot\_idx$
```

4) POOLING LAYER

As max-pooling is a non-polynomial function, a polynomial function is needed to replace the max-pooling function. The HE scheme only supports multiplication and not division. Note that f is the size of the filter in the pooling layer.

However, the division by $f \times f$ can be considered as multiplication by $\frac{1}{f \times f}$, as in (1). Thus, to modify the max-pooling layer into the average pooling layer, we adopt the approach of multiplying the $\frac{1}{f \times f}$ the inverse of the number of the elements of the filter directly over all the elements of the filter. Note that the multiplication is performed over the ciphertext without decryption.

$$avg() = \frac{1}{f \times f} \sum_{i=1}^{f \times f} x_i \longrightarrow \sum_{i=1}^{f \times f} \frac{1}{f \times f} x_i \tag{1}$$

where the mask vector consists of 0 and $\frac{1}{f \times f}$ and the MD of the average-pooling layer is 1.

5) BATCH NORMALIZATION

To ensure the stability of the NN training, it is necessary to carefully select the initialization method and choose a small value of learning rate, despite it increasing the complexity of the task. To address the limitation, the Google team proposed the BN [11] method to help train the network better. Average μ and variance σ^2 of elements of the BN layer in mini-batch training are used to normalize elements into a fixed range, subjected to natural distribution. Considering the effect of the normalization on performance, the authors implemented two learnable parameters γ and β, to scale and shift values, respectively.

Because of the leveled HE scheme, the inference latency of the PPDL model is highly related to the predefined level. Deploying the BN layer increases the MD, which sets a higher level and leads to longer latency. Since the BN layer is adopted into the typical NN model and then transformed for the NN model for ciphertext inference, its parameters μ, variance σ^2, scale γ, and shift β are deployed with multiplication. These four parameters are implemented in several arithmetic operations, which increases the MD. The CM is a required tool to reduce MD. It focuses on coalescing parameters used in the BN layer with the other layers, such as the activation layer. Ioffe [11] *et al.* deployed the BN layer after each activation layer to improve accuracy. We pointed out two schemes of fusing the BN and activation layer in the encrypted model, one is Convolution-Activation-BN, which we call the "CAB" scheme, and the other is Convolution-BN-Activation, which we call the "CBA" scheme. We develop the fused layer via CM as the "Mapping" layer, described in detail in Section III-B.

6) PACK LAYER

The ciphertexts are the chaotic vectors that must be rearranged before feeding feature maps into the Fully Connected (FC) layer. Due to the vector-matrix multiplication in the FC layer, the model needs to pack these ciphertexts into one ciphertext whose valid slots must be equal to the input size of the following FC layer; otherwise, the model is unable to perform the HE addition. The required method is flattening the valid slots of all ciphertexts into one ciphertext. However, there are two problems with this method. Firstly, ciphertext has redundant slots that do not casually concatenate the rear with the head. Secondly, each ciphertext is chaotic, which means that the valid slots of ciphertext are not neighbors.

The pack layer concatenates all the valid slots, which are located along the length of the flattened channel vector. The length of slots of one ciphertext is fixed, whose size equals $\frac{N}{2}$ (half of the polynomial degree N); it is not feasible to connect the end of one ciphertext to the head of another ciphertext, as it would exceed the maximum length. Thus, concatenating slots equal to the flattened channel's length rather than the ciphertext length would be a better alternative. However, there is a problem. If the length or the number of ciphertexts is too large, the concatenated slots will exceed the specified length, which implies that the valid slots are sparse. So, a well-structured output vector is needed in the FC layer. We propose the algorithm-Flatten Multi-ciphertext to Single-ciphertext (FM2S) to pack ciphertexts into a single ciphertext, written down as Algorithm 4.

Algorithm 4 Flatten Multi-Ciphertexts to Single-Ciphertext (FM2S)

```
Input: input feature map $[x]$, last output feature map size $n$, valid list $v$, mask $m$
Output: packed ciphertext $[x\_pac]$
1: Obtain channels of $[x]$ : $c \longleftarrow len([x])$
2: for $i = 1, \ldots, n^2 - 1$ do
3:     $rot\_idx_i \longleftarrow v_i - i$
4: end for
5: for $j = 0, \ldots, c - 1$ do
6:     $[y] \leftarrow [x_j]$
7:     $mask \leftarrow Encode(m)$
8:     $[Z_0] \leftarrow HEmult([y], mask)$
9:     for $k = 0, \ldots, n^2 - 1$ do
10:        $mask_{v_{k+1}} \leftarrow 1$
11:        $[Z_{k+1}] \leftarrow Rot(HEmult([y], mask), rot\_idx_k)$
12:    end for
13:    $[y] \leftarrow HEadd([Z_0], \ldots, [Z_{n^2-1}])$
14:    $[Y_i] \leftarrow Rot([y], -(n \times n \times i))$
15: end for
16: $[x\_pac] \leftarrow HEadd([Y_0], \ldots, [Y_{c-1}])$
17: return $[x\_pac]$
```

7) FULLY CONNECTED LAYER

For the FC layer of the NN model, the filter is a matrix of size (input channel × output channel), and it involves matrix multiplication of the input and the filter matrix. In privacy-preserving CNN, before feeding feature maps into the FC layer, feature maps should be flattened into one ciphertext in the mentioned pack layer, which shapes a single vector. The input and the output of the FC layer are presented in a single ciphertext. However, the ciphertext encrypted by HE does not support matrix multiplication. Consequently, we estimated using HE rotation and HE multiplication to execute the matrix multiplication function. We change the matrix-vector multiplication scheme into a vector-matrix multiplication scheme. We leverage the approach proposed by Halevi and Shoup [28], which is a multiplication operation between vector and tensor (matrix).

The first phase of this layer is to preprocess the weight matrix. Since the output channel is smaller than the input channel, it is crucial to pad the matrix by 0 to the square matrix of size "input channel × input channel." Next, the padded matrix is diagonalized along with the raw. Vector-matrix multiplication performs element-wise multiplication between the ciphertext and plaintext weight matrix, which can only be achieved by HE rotation.

We found that the vector-matrix multiplication has an unnecessary step, which increases latency. Assuming the input channel is I, and the output channel is O, it needs $2(I-1)$ times HE rotations and one time HE addition, where there are $\frac{2(I-1)}{2}$ times HE rotation to the left and $\frac{2(I-1)}{2}$ times HE rotation to the right linearly. To reduce the HE operation, we propose the method that implements fewer than $\frac{2(I-1)}{2}$ times HE rotation to the right linearly, as shown in Algorithm 5. The instance is presented in Fig. 5.

In Fig. 5, elements of the part η are computed as 0 in the output, which implies that the elements do not affect the output. Currently, as the HE rotation is a linear operation, intersection $I_{\delta,\eta}$ of the part δ and part η are in serial, which means that the HE rotation can assemble linear transformation simultaneously. On the other hand, intersection $I_{\zeta,\eta}$ of the part ζ and η are not in serial, which means the transformation is non-linear, and two HE rotations are necessary. The previous method costs $2(I-1)$ HE rotations in the FC layer. As for the proposed enhanced approach, the intersection $I_{\delta,\eta}$ requires $I-O$ times HE rotations, which runs one time HE rotation to each line; the intersection $I_{\zeta,\eta}$ expends $2(I-(I-O)-1) = 2(O-1)$ times HE rotations, which executes two HE rotations to each line. The number of HE rotations is changed to $I+O-2$, i.e., the latency of this step has become $\frac{I+O-2}{2(I-1)}$ of the original. Thus for the aforementioned example, which demonstrated that it is possible to ignore a part of the HE rotation moving to the right, the latency becomes $\frac{7+3-2}{2(7-1)} = \frac{2}{3}$ of the original.

The relationship between the input channel and the output channel is drawn in Fig. 6. The x- and y- axes represent the FC layer's input and output channels. The z-axis is the ratio (%) of the number of HE operations of original and improved methods. Before implementing the improved algorithm, the HE operation of the FC layer needs to be performed $2(I-1)$ times. After that, the number of HE operations becomes $I+O-2$ times. This ratio implies that the latency of the FC layer can be reduced to somewhat the original by using an improved algorithm, depending on the number of input and output channels. A lower ratio means better results for the improved algorithm.

Algorithm 5 Vector-Matrix Multiplication (VMM)

Input: input feature map $[x]$, weight matrix w, bias b
Output: ciphertext $[x']$

```
1:  $[t] \leftarrow [x]$
2:  Obtain input feature map size $m \leftarrow len(w)$
3:  Obtain output feature map size $n \leftarrow len(b)$
4:  Set boundary $e \leftarrow m - n + 1$
5:  $w \leftarrow Diagonalization(Padding(w))$
6:  for $i = 0, \dots, m-1$ do
7:      $w_i \leftarrow Encode(w_i)$
8:      $[Y_i] \leftarrow HEmult([t], w_i)$
9:      IF $i \neq 0$ or $i < e$:
10:         $[t] \leftarrow Rot([x], i+1)$
11:     IF $i \neq m-1$ and $i \geq e$ :
12:         $[t] \leftarrow HEadd(Rot([x], i+1), Rot([x], i+1-m))$
13: end for
14: $[Y] \leftarrow HEadd([Y_0], \dots, [Y_{m-1}])$
15: $[Y] \leftarrow Rescale([Y])$
16: $B \leftarrow Encode(b)$
17: $[x'] \leftarrow HEadd([Y], B)$
18: return $[x']$
```

B. BATCH NORMALIZATION WITH COEFFICIENT MERGING

To improve the accuracy and reduce the MD of ciphertext inference in CNN, we process the BN layer with CM. By employing qualitative modes of inquiry, we attempt to illuminate the two different schemes in mathematics. The derivations of mathematical formulae of the schemes are shown as the following.

The formulae of the two schemes are described in order. A convolution layer can be written down as in (2), where $[I_f]$ is the input feature map, $f_{11}, \dots, f_{1f}, \dots, f_{ff}$ are weight elements of the filter of size $f \times f$, and $[X_{conv}]$ is the output of *rotation & accumulation*. Furthermore, $[Y_{conv}]$ can be obtained by the *mask computation*, which leverages a mask m to remove the redundant slots "#," as in (3). The process of obtaining $[X_{conv}]$ and $[Y_{conv}]$ are shown in Fig. 3.

$$[X_{conv}] = [I_f]f_{11} + \cdots + rot([I_f], f \times f - 1)f_{ff} \quad (2)$$
$$[Y_{conv}] = m[X_{conv}] \quad (3)$$

1) CONVOLUTION-ACTIVATION-BN (CAB) SCHEME

A second-order polynomial is used to approximate the activation function, as in (4). In the BN layer, the normalized $[\overline{Y_{act}}]$ is calculated by the input $[Y_{act}]$, the mean μ and variance σ^2. In addition, due to the gradient dispersion problem, the output is multiplied by the scale γ (weight), and the shift β (bias) is also added. After simplifying the formula, the actual second-order polynomial approximation is expressed as in (5). We have replaced the activation function with the square function, which implies that the coefficients a, b, and c of the second-order function are 1, 0, and 0, respectively. The mask

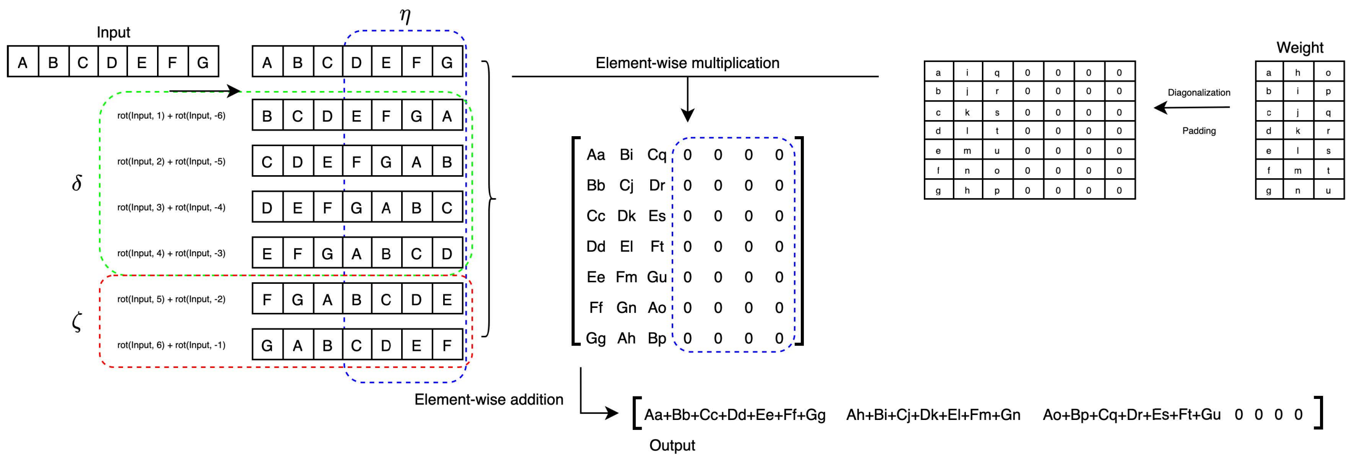

Input
A B C D E F G
η
δ
ζ
rot(Input, 1) + rot(Input, -6)
rot(Input, 2) + rot(Input, -5)
rot(Input, 3) + rot(Input, -4)
rot(Input, 4) + rot(Input, -3)
rot(Input, 5) + rot(Input, -2)
rot(Input, 6) + rot(Input, -1)
Element-wise multiplication
Element-wise addition
Aa+Bb+Cc+Dd+Ee+Ff+Gg Ah+Bi+Cj+Dk+El+Fm+Gn Ao+Bp+Cq+Dr+Es+Ft+Gu 0 0 0 0
Output
Diagonalization
Padding
Weight

FIGURE 5. Vector-matrix multiplication. Assume the input is ciphertext with the first seven valid elements [A B C D E F G], which has a shape of a 1 × 7 vector, and the filter is 7 × 3 matrix which implies that the input channel is seven and the output channel is three of the FC layer.

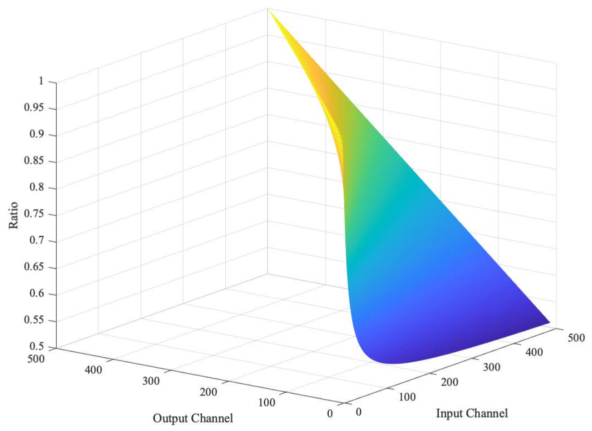


FIGURE 6. The relationship between the input channel and the output channel of the FC layer.

m used in the convolution layer is a vector containing only 0 and 1. Its objective is to obtain a result without the redundant slots "#." So the value of m is regarded as 1 logically, i.e., the value of m^2 is regarded as 1. In conclusion, for the specific example of a square function, the output feature map $[Y_{bn}]$ of the CAB scheme is changed into $[Y'_{bn}]$, as in (6). After reducing the MD, as proposed by Ishiyama *et al.* [4], the output of the CAB scheme is formulated as in (6).

$$[Y_{act}] = a[Y_{conv}^2] + b[Y_{conv}] + c \tag{4}$$

$$\begin{aligned}[Y_{bn}] &= \gamma[\overline{Y_{act}}] + \beta \\ &= \frac{\gamma a m^2}{\sqrt{\sigma^2+\epsilon}}[X_{conv}^2] + \frac{\gamma b m}{\sqrt{\sigma^2+\epsilon}}[X_{conv}] \\ &\quad + (\beta - \frac{\gamma(\mu - c)}{\sqrt{\sigma^2+\epsilon}}) \end{aligned} \tag{5}$$

$$\begin{aligned}[Y'_{bn}] &= \frac{\gamma}{\sqrt{\sigma^2+\epsilon}}[X_{conv}^2] + (\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}) \\ &= [X_{conv}^2] + B \end{aligned} \tag{6}$$

$$where \quad B = (\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})/\frac{\gamma}{\sqrt{\sigma^2+\epsilon}} \tag{7}$$

2) CONVOLUTION-BN-ACTIVATION (CBA) SCHEME

In the BN layer, the normalized $[\overline{Y_{conv}}]$ is calculated by the mean μ and variance σ^2 with the input $[Y_{conv}]$. In addition, $[\overline{Y_{conv}}]$ also needs to be multiplied by the scale γ and added with the shift β; the resulting well-structured formula is described in (8). For the CBA scheme, the output feature map of the BN layer is followed by the activation layer, as in (10). Furthermore, (10) substituted $[Y_{conv}]$ with $[X_{conv}]$ combined within (2). For the specific example of a square function, the output feature map $[Y_{act}]$ of the CBA scheme is changed into $[Y'_{act}]$, after CM, as in (11). After reducing the MD, as proposed by [4], the output of the CBA scheme is formulated as in (12).

$$[Y_{bn}] = \gamma[\overline{Y_{conv}}] + \beta \tag{8}$$

$$[Y_{act}] = a[Y_{bn}^2] + b[Y_{bn}] + c \tag{9}$$

$$\begin{aligned} &= \frac{a\gamma^2 m^2}{\sigma^2+\epsilon}[X_{conv}^2] + [2a(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})\frac{\gamma}{\sqrt{\sigma^2+\epsilon}} \\ &\quad + \frac{b\gamma}{\sqrt{\sigma^2+\epsilon}}]m[X_{conv}] + [a(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})^2 \\ &\quad + b(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}) + c] \end{aligned} \tag{10}$$

$$\begin{aligned}[Y'_{act}] &= \frac{\gamma^2}{\sigma^2+\epsilon}[X_{conv}]^2 + [2(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}) \\ &\quad \times \frac{\gamma}{\sqrt{\sigma^2+\epsilon}}][X_{conv}] + (\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})^2 \end{aligned} \tag{11}$$

$$= [X_{conv}^2] + C[X_{conv}] + D \tag{12}$$

$$\begin{aligned} where \quad C &= [2(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}]/\frac{\gamma^2}{\sigma^2+\epsilon} \\ D &= (\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}})^2/\frac{\gamma^2}{\sigma^2+\epsilon} \end{aligned}$$

The MD of the mapping layer becomes one after CM and optimization. The BN and activation layer are separate when

TABLE 3. The network architectures of PPNNs. C, A, P, B, and F denote convolution, activation, pooling, batch normalization, and fully-connected layers.

Network	Architecture
CryptoNets [1]	C-A-P-A-F
Light CNN [2]	C-P-C-P-F-B-A-F
HCNN [6]	C-A-C-A-F
CNN A [3]	C-P-C-A-P-F-F
CNN B [4]	C-B-A-C-B-A-F

the NN model is deployed over plaintext, but they are fused into one, the mapping layer, when deployed over ciphertext, to optimize the MD. The mapping layer produces completely different outputs for the two schemes when applied to the output of the convolution layer. The CAB scheme produced a second-degree term with one constant, while the CBA scheme produced an additional first-degree term. These different mappings result in their disparate accuracy and latency.

IV. EXPERIMENT

To verify if the latency of encryption/decryption for CHE is lower than that for PiHE, we have compared their performances for one instance. Furthermore, we have applied the CHE to five previous NN architectures [1], [2], [3], [4], [6] and measured the accuracy and latency of ciphertext inference for comparison.

A. DATASETS AND NETWORKS

1) DATASETS

We adopt the MNIST [5] and the CIFAR-10 [29] datasets to evaluate our proposed methods. We used the MNIST and the CIFAR-10 datasets as the same datasets used in previous papers. Both datasets are standard datasets in this research area. The MNIST is a dataset of handwritten digits, which has a training set of $60,000$ images and a test set of $10,000$ images, each of size 28×28, divided into 10 classes of numbers from 0 to 9. It is a gray image dataset, where each image has a single channel. The CIFAR-10 dataset consists of $60,000$ color images, each of the size 32×32 in 10 classes, with $6,000$ images per class. There are $50,000$ training images and $10,000$ test images. Unlike the former, each image in CIFAR-10 has three channels.

2) NETWORKS

We adopted the CNN to evaluate performance and compare the five networks shown in Table 3 with the proposed CHE. Initially, these studies adopted the pixel-wise approach. We replicated their architecture precisely and applied the CHE and CM schemes to improve their performance.

B. EXPERIMENTAL SETUP

We ran all comparison models on the same Linux server in parallel, equipped with Intel Xeon E5-2660 with 126 GB memory and 40 cores. We only focused on the ciphertext inference stage and used Pytorch to train all the NN models with plaintext. The epoch was set as 100, Adam ($\beta_1 = 0.9, \beta_2 = 0.99$) as the optimization, the learning rate as 0.01. We adopted the Microsoft SEAL 3.7 [30] in Python 3.9 using Python-SEAL [31] to encrypt and decrypt data with the RNS-CKKS HE scheme [22]. All PPDL models in our experiments could achieve the $128-$bit security level.

To simulate an actual situation when the user sends one query to the server, we set the inference image of every epoch to be one. Furthermore, HE flattens each image channel into one vector and encrypts it into ciphertext. For example, each image of CIFAR-10 has three channels flattened into three vectors and encrypted as three ciphertexts. So, the model gets a list, with three ciphertexts, as the input to HE-friendly CNN. After running over the model, the model's output is a single ciphertext with valuable front parts (10 slots) and redundant rear parts (other slots). The user cuts down decrypted output and feeds it into the Softmax function to obtain the prediction. The convolution layer, the pack layer, and the FC layer are the three tough layers in the model used for parallel computing; serial calculation is used for the rest.

C. RESULT

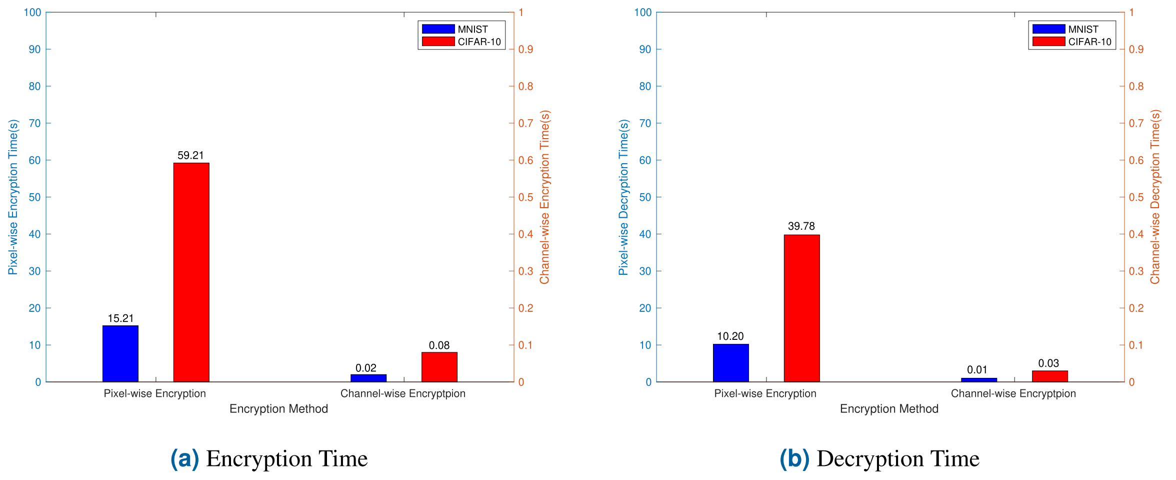
1) ENCRYPTION AND DECRYPTION

We first evaluated the time required for encryption and decryption in the CHE and PiHE. This experiment encrypts the same instance into ciphertext using the CHE and PiHE, respectively, and then decrypts ciphertext into plaintext, counting the encryption and decryption time. The encryption and decryption run time depends on the number of ciphertexts.

The aim was to validate if the former takes less encryption/decryption time than the latter. A random instance was chosen from the MNIST and CIFAR-10, and the example image was processed using the two approaches while logging the encryption and decryption times. The test was performed 100 times to obtain an average result, shown in Fig. 7. Statistically, under datasets, MNIST and CIFAR-10, deploying with the CHE cost 0.02 seconds and 0.08 seconds for encryption and cost 0.01 seconds and 0.03 seconds for decryption, respectively. On the contrary, the PiHE took 15.21 seconds and 59.21 seconds for encryption and 10.20 seconds and 39.78 seconds for decryption, respectively. As expected, the CHE took a significantly shorter time for encryption and decryption.

2) COEFFICIENTS MERGING SCHEMES

To test the effect of different schemes on the performance, a simple model HCNN [6] and dataset MNIST were chosen. As shown in Table 4, CAB and CBA obtained an accuracy of 98.50% and 99.03%, respectively, for ciphertext inference. The attained latency was 10.35 seconds and 10.96 seconds, respectively. The experimental results reveal that CBA has higher accuracy and longer latency, which conforms with the formulae we derived in Section III-B.



(a) Encryption Time **(b)** Decryption Time

FIGURE 7. Encryption and decryption latency on the datasets MNIST and CIFAR-10 datasets for one instance.

TABLE 4. The accuracy and latency of different schemes.

Dataset	Model	scheme	Ciphertext Infer. Accuracy (%)	Infer. Latency (second)
MNIST	HCNN	CAB	**98.50**	**10.35**
		CBA	**99.03**	**10.96**

Due to the fused layer output, the results of the CBA scheme were stable with high accuracy, but the latency became a bit long. In contrast, as the CAB scheme does not have enough lower-order terms to analyze the details of the feature map, certain elements were ignored during inference. The CAB scheme resulted in a shorter latency but reduced accuracy.

3) ACCURACY AND LATENCY

Accuracy is the primary performance measuring a parameter of the inference. As the CBA scheme achieved higher accuracy than the CAB scheme, thus, we performed the CHE with the CBA scheme to verify if the CHE implementation improved the accuracy and latency on the MNIST and CIFAR-10. The ciphertext inference time was logged, and all recorded values were the average of 10 times experiments.

Table 5 shows the results of the five models and compares the accuracy and latency of the original models and the proposed model on the MNIST. The proposed method leverages the CHE to process the whole model, and the fused layer employs the CBA scheme. It achieved an accuracy of 99.00% for the CryptoNets [1], 98.05% for the light CNN [2], 99.30% for the HCNN [6], 99.32% for the CNN A [3], and 99.30% for the CNN B [4]. The corresponding latency values for the five models were 15.22 seconds, 17.89 seconds, 10.96 seconds, 16.14 seconds, and 7.76 seconds, respectively. Furthermore, the speedups are 37.45, – (no previous inference time), 1.29, 23.96, and 2.73, respectively.

The results showed better performance than the previous five works, especially speedup. As only the work of CNN B evaluated the CIFAR-10 dataset, Table 6 presented the single result of the CNN B model and compared it with the accuracy and latency of the original and proposed models on CIFAR-10. Our method leverages the CHE to process all models, and the BN layer employs the CBA scheme. The proposed model achieved an accuracy of 76.40%, latency of 111.91 seconds, and speedup of 9.28. Thus, the proposed model establishes better results for this case too.

V. DISCUSSION

A. EXPLANATION

To perform inference on large datasets in the NN model, researchers packed the same pixel of multiple images in a single ciphertext, which could test many ciphertext images in one inference, called the PiHE. Previous research [3], [6], [25] have shown reasonable accuracy and latency in interactive and non-interactive paradigms.

The PiHE uses tensors to process data, and each element of a tensor is ciphertext. The CHE leverages data upon ciphertext; the data is ciphertext rather than plaintext. Dathathri *et al.* [9] and Lou *et al.* [10] implemented a similar CHE presented in their figures. However, it is challenging to derive their exact algorithm from the figures in which it has been presented. Also, their method is used only for the convolution layer, while our proposed algorithm applies to all kinds of layers in the NN, including the convolution layer, pooling layer, activation layer, BN layer, and FC layer.

Consistent with previous studies, this paper shows that the CHE can achieve higher accuracy and shorter latency than current PPDL models under multi-processing. Applying the CHE to encrypt data into ciphertext in the number of channels, we found that the proposed HE-friendly algorithms for the CHE in CNN are prudent and reliable. The algorithm is predominantly expressed in different layers and easily exploited to transform NN models. As for the generalization of the CHE, since the algorithm inside the model has been adapted to the ciphertext calculation, it is possible to transform any neural network model using the CHE. At the same

TABLE 5. The inference latency and accuracy of previous models on MNIST. A multi-processing situation evaluates latency. *L* is the level of the model, Scale is the scale factor of HE, *N* is the polynomial degree, and *Q* is modulo.

Network	L	Scale (b)	N (b)	Q (b)	PiHE/Infer. Acc. in paper (%)	**CHE/Proposed method Acc.** (%)	PiHE/Infer. Time in paper (second)	**CHE/Proposed method Infer. Time (second)**	**Speedup**
CryptoNets	7	2^{30}	16384	330	98.95 [1]	**99.00 (+0.05)**	570.00 [1]	**15.22 (−554.78)**	**37.45**
light CNN	10	2^{30}	16384	420	97.91 [2]	**98.05 (+0.14)**	-	**17.89 (-)**	**-**
HCNN	8	2^{30}	16384	360	99.00 [6]	**99.30 (+0.30)**	14.11 [6]	**10.96 (−3.15)**	**1.29**
CNN A	10	2^{30}	16384	420	99.25 [3]	**99.32 (+0.07)**	386.70 [3]	**16.14 (−370.56)**	**23.96**
CNN B	8	2^{30}	16384	360	99.18 [4]	**99.30 (+0.12)**	21.15 [4]	**7.76 (−13.39)**	**2.73**

TABLE 6. The inference latency and accuracy of previous model CNN B on CIFAR-10. *L* is the level of the model, Scale is the scale factor of HE, *N* is the polynomial degree, and *Q* is modulo.

Network	L	Scale (b)	N (b)	Q (b)	PiHE/Infer. Acc. in paper (%)	**CHE/Proposed method Acc.** (%)	PiHE/Infer. Time in paper (second)	**CHE/Proposed method Infer. Time (second)**	**Speedup**
CNN B	8	2^{30}	16384	360	76.37 [4]	**76.40 (+0.03)**	1038.00 [4]	**111.91 (−926.09)**	**9.28**

time, for the input data, it is only necessary to transform it into vector data encrypting by homomorphic encryption for ciphertext inference.

The CAB and CBA schemes show only minor significant differences in accuracy and latency; it must be pointed out that, due to the CAB formula, only the profile of the image is used for the results, lowering the scheme's accuracy. On the contrary, due to the CBA formula, the image profile and other details are used for the results, leading to a higher frequency but longer latency. Because there are different output polynomial terms in the fused layer, the CAB scheme is more suitable for quick inference over ciphertext, where the focus is on central architecture; and the CBA scheme is more suitable for precise inference over ciphertext to not only see the main body but also pay attention to the details. We believe that the minor differences in the accuracy and latency were also because of the implementation of the second-order square function. Nevertheless, the final formulae are not hugely different after deploying CM through the proposed CAB and CBA schemes because the CBA has only one additional first-order term using the square function. If higher-order polynomials are deployed, the final formulae after CM present more significant differences, increasing the variation in the accuracies and latencies. Technically, the CBA scheme provides a more stable analysis in ciphertext inference, and the CAB scheme provides a quicker analysis.

B. LIMITATION

Despite the preliminary character of CHE, this study indicates that reasoning and corroborating from the fundamentals of data structures can prove its effectiveness and efficiency. However, there is a restriction on inference latency due to the sophistication and black-box nature of ciphertext. In addition, the differences in the accuracies and latencies of the CBA and CAB could be significant if a function more sophisticated than the square function is used. Since there is no source code to reproduce the previous works, we leave the direct comparison with the previous studies under a specific platform as a future study.

C. SIGNIFICANCE

Future iterations of inference over ciphertext from CHE to instance-wise encryption, which encrypts one instance as the single ciphertext, may demonstrate even greater potency. Hopefully, the experimental results improve inference over ciphertext, significantly changing the basic data structures used for PPDL inference.

VI. CONCLUSION

This study sets out to improve the performance of HE-based PPDL by combing the proposed approaches of CHE and the BN layer with CM. It also discusses several related algorithms in detail and the computation scheme "Onion." The CHE implements ciphertext inference for the end-user in the non-interactive paradigm. The BN layer with CM improved the accuracy and decreased the latency by reducing the MD, using two proposed schemes, the CBA and CAB. The proposed method achieved the highest accuracy of 99.32% and the shortest latency of 7.76 seconds on the MNIST dataset compared to five previous architectures. It also attained an accuracy of 76.4% and a latency of 111.91 seconds on the CIFAR-10. Thus, our experiments demonstrate that the CHE can serve as a tool to design a more robust and flexible PPDL model which performs ciphertext inference in the CNN with better accuracy and latency. In future work, we will target more challenging problems with processing on graphical processing units, actual datasets, and deeper networks and aim to achieve lower latency and higher accuracy for instance-wise ciphertext inference that encrypts one instance as the single ciphertext.

REFERENCES

[1] 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 *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 2016, pp. 201–210.

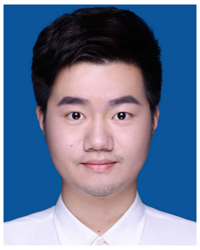
[2] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," Cryptol. ePrint Arch., IACR, Lyon, France, Tech. Rep. 2017/035, 2017. [Online]. Available: https://eprint.iacr.org/2017/035

[3] E. Hesamifard, H. Takabi, and M. Ghasemi, "Deep neural networks classification over encrypted data," in *Proc. 9th ACM Conf. Data Appl. Secur. Privacy*, Baltimore, MD, USA, Mar. 2019, pp. 97–108.

[4] T. Ishiyama, T. Suzuki, and H. Yamana, "Highly accurate CNN inference using approximate activation functions over homomorphic encryption," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 3989–3995.

[5] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "TenSEAL: A library for encrypted tensor operations using homomorphic encryption," 2021, *arXiv:2104.03152*.

[6] A. Al Badawi, C. Jin, J. Lin, C. F. Mun, S. J. Jie, B. H. M. Tan, X. Nan, K. M. M. Aung, and V. R. Chandrasekhar, "Towards the AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1330–1343, Jul. 2021, doi: 10.1109/TETC.2020.3014636.
[7] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30039–30054, 2022.
[8] E. Aharoni, A. Adir, M. Baruch, N. Drucker, G. Ezov, A. Farkash, L. Greenberg, R. Masalha, G. Moshkowich, D. Murik, H. Shaul, and O. Soceanu, "HeLayers: A tile tensors framework for large neural networks on encrypted data," 2020, *arXiv:2011.01805*.
[9] R. Dathathri, O. Saarikivi, H. Chen, K. Laine, K. Lauter, S. Maleki, M. Musuvathi, and T. Mytkowicz, "CHET: An optimizing compiler for fully-homomorphic neural-network inferencing," in *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, New York, NY, USA, Jun. 2019, pp. 142–156.
[10] Q. Lou and L. Jiang, "HEMET: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 7102–7110.
[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, Lille, France, Jul. 2015, pp. 448–456.
[12] J. Chiraag, V. Vinod, and C. Anantha, "GAZELLE: A low latency framework for secure neural network inference," in *Proc. USENIX Secur. Symp. (USENIX Secur.)*, Baltimore, MD, USA, Aug. 2018, pp. 1651–1669.
[13] Q. Zhang, C. Wang, H. Wu, C. Xin, and T. Phuong, "GELU-Net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning," in *Proc. IJCAI*, Stockholm, Sweden, Jul. 2018, pp. 3933–3939.
[14] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE SP*, San Jose, CA, USA, May 2017, pp. 19–38.
[15] M. S. Riazi, C. Weinert, O. Tkachenko, M. E. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. Asia Conf. Comput. Commun. Secur.*, May 2018, pp. 707–721.
[16] J. Liu, Y. Tian, Y. Zhou, Y. Xiao, and N. Ansari, "Privacy preserving distributed data mining based on secure multi-party computation," *Comput. Commun.*, vol. 153, pp. 208–216, Mar. 2020.
[17] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," 2016, *arXiv:1610.05755*.
[18] W. Fan, J. He, M. Guo, P. Li, Z. Han, and R. Wang, "Privacy preserving classification on local differential privacy in data centers," *J. Parallel Distrib. Comput.*, vol. 135, pp. 70–82, Jan. 2020.
[19] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, and J. Li, "A training-integrity privacy-preserving federated learning scheme with trusted execution environment," *Inf. Sci.*, vol. 522, pp. 69–79, Jun. 2020.
[20] K. Kim and H. Tanuwidjaja, *Privacy-Preserving Deep Learning A Comprehensive Survey*. Cham, Switzerland: Springer, 2021.
[21] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st ACM Symp. Theory Comput. (STOC)*, Bethesda, MD, USA, May 2009, pp. 169–178.
[22] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Hong Kong, Nov. 2017, pp. 409–437.
[23] P. Xie, B. Wu, and G. Sun, "BAYHENN: Combining Bayesian deep learning and homomorphic encryption for secure DNN inference," 2019, *arXiv:1906.00639*.
[24] T. Xie and Y. Li, "Efficient integer vector homomorphic encryption using deep learning for neural networks," in *Proc. Int. Conf. Neural Inf. Process.*, Siem Reap, Cambodia, Dec. 2018, pp. 83–95.
[25] J. Lee, E. Lee, J.-W. Lee, Y. Kim, Y.-S. Kim, and J.-S. No, "Precise approximation of convolutional neural networks for homomorphically encrypted data," 2021, *arXiv:2105.10879*.
[26] P. Andrew, M. Andrew, and B. Ian, "Modelling and automatically analysing privacy properties for honest-but-curious adversaries," Univ. Oxford, Oxford, U.K., Tech. Rep., 2014.
[27] R. E. Ali, J. So, and A. S. Avestimehr, "On polynomial approximations for privacy-preserving and verifiable ReLU networks," 2020, *arXiv:2011.05530*.
[28] S. Halevi and V. Shoup, "Algorithms in HElib," in *Proc. Annu. Int. Cryptol. (CRYPTO)*. Santa Barbara, CA, USA, Aug. 2014, pp. 554–571.
[29] K. Alex and H. Geoffrey, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[30] C. Hervé, D. Amaury, M. Jonathan, M. Constance, and P. Emmanue. (2021). Microsoft SEAL (release 3.7). Microsoft Research. Redmond, WA, USA. [Online]. Available: https://github.com/Microsoft/SEAL
[31] *Seal-Python. Huelse*. Accessed: Jun. 2020. [Online]. Available: https://github.com/Huelse/SEAL-Python



TIANYING XIE (Student Member, IEEE) received the M.Sc. degree from the College of Computer and Information Science, Southwest University, Chongqing, China, in 2020. He is currently pursuing the Ph.D. degree in computer science and communications engineering with Waseda University. His research interests include privacy-preserving deep learning, homomorphic encryption, differential privacy, and federated learning.

HAYATO YAMANA (Member, IEEE) has been a Professor with the Faculty of Science and Engineering, Waseda University, since 2005. He is currently a member of ACM and a Senior Member of IEICE. He served on the Board of Governors of the IEEE Computer Society, from 2018 to 2020.

TATSUYA MORI (Member, IEEE) received the B.E. and M.E. degrees in applied physics and the Ph.D. degree in information science from Waseda University, Tokyo, Japan, in 1997, 1999, and 2005, respectively. He joined at the NTT Laboratory, in 1999. Since then, he has been engaged in the research of measurement and analysis of networks and cyber security. From March 2007 to March 2008, he was a Visiting Researcher at the University of Wisconsin–Madison. He is currently a Professor with Waseda University. He received the Telecom System Technology Award from TAF, in 2010, and the Best Paper Awards from IEICE and IEEE/ACM COMSNETS, in 2009 and 2010, respectively. He is a member of ACM, IEICE, IPSJ, and USENIX.