# GPU Accelerated Full Homomorphic Encryption Cryptosystem, Library and Applications for IoT Systems

Xiaodong Li, Hehe Gao, Jianyi Zhang, Shuya Yang, Xin Jin, and Kim-Kwang Raymond Choo, *Senior Member, IEEE*

***Abstract*—Deep learning such as convolutional neural networks (CNNs) have been utilized in a number of cloud-based Internet of Things (IoT) applications. Security and privacy are two key considerations in any commercial deployments. Fully homomorphic encryption (FHE) is a popular privacy protection approach, and there have been attempts to integrate FHE with CNNs. However, a simple integration may lead to inefficiency in single-user services and fail to support many of the requirements in real-time applications. In this paper, we propose a novel confused modulo projection based FHE algorithm (CMP-FHE) that is designed to support floating-point operations. Then we developed a parallelized runtime library based on CMP-FHE and compared it with the widely employed FHE library. Our results show that our library achieves a faster speeds. Furthermore, we compared it with the state-of-the-art confused modulo projection based library and the results demonstrated a speed improvement of 841.67 to 3056.25 times faster. Additionally, we construct a Real-Time Homomorphic Convolutional Neural Network (RT-HCNN) under the ciphertext-based framework using CMP-FHE, as well as using graphics processing units (GPUs) to facilitate acceleration. To demonstrate utility, we evaluate the proposed approach on the MNIST dataset. Findings demonstrate that our proposed approach achieves a high accuracy rate of 99.13%. Using GPUs acceleration for ciphertext prediction results in us achieving a single prediction time of 79.5 ms. This represents the first homomorphic CNN capable of supporting real-time application and is approximately 58 times faster than Microsoft's Lola scheme.**

***Index Terms*—Convolutional neural networks (CNNs), Fully homomorphic encryption (FHE), Internet of Things (IoT), Graphics processing units (GPUs).**

## I. Introduction

DEEP learning techniques (e.g. convolutional neural networks - CNNs), have been widely deployed in a broad range of applications, including cloud-based Internet of Things (IoT) systems. Typically such an approach employs artificial neural networks to extract high-level abstract features through multi-layer nonlinear transformations. By training on large-scale datasets, these networks have the capability to acquire a deep understanding of intricate patterns and relationships, as evidenced in applications like image recognition, speech recognition, natural language processing, and recommender systems.

In the context of cloud-based IoT systems, such systems and connected devices (e.g., sensors) generally capture large volumes of structured or unstructured data (e.g., images, audio, and text), which can then be processed by CNNs in the cloud. In other words, the deployment of CNNs in cloud-based IoT systems enables real-time data processing and intelligent decision-making, thereby enhancing the overall performance and intelligence of such systems. However, such a workflow carries inherent risks. In Fig. 1, we can observe an example of potentially sensitive information leakage. This highlights the significance of AI services deployed on the cloud within the IoT business landscape. With the widespread adoption of IoT data collection devices, gaining a comprehensive understanding of the potential risks associated with gathering users' personal and private data is imperative. There exists a possibility that such data could be vulnerable to exploitation by unscrupulous individuals during both the transfer process and its storage on cloud services. Meanwhile, users are often reluctant to hand over their sensitive data directly to the cloud server to prevent malicious use by the service provider. Addressing these security concerns proactively is paramount in ensuring data privacy and protection in IoT environments. If a traditional encryption scheme is used, the cloud server cannot directly process the user's encrypted private data without decrypting it. A natural question is how do we preserve data privacy and security while facilitating real-time intelligence sharing and maximizing data utility.

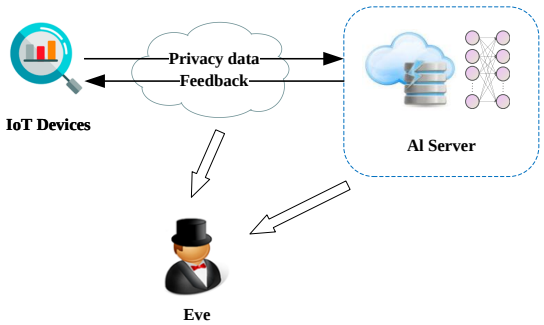


Fig. 1. IoT device privacy data leakage risk model

Full homomorphic encryption is considered to be one of the popular ways to solve the above problem, which can

Xiaodong Li, Hehe Gao, Jianyi Zhang, Shuya Yang and Xin Jin are with Department of Cyber Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China.

Corresponding author: Jianyi Zhang, zjy@besti.edu.cn.

Kim-Kwang Raymond Choo is with Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA.

be computed directly without decrypting the ciphertext, and the decrypted result of the ciphertext obtained is consistent with the result of the same computation performed on the plaintext, and it has been widely used in the field of Internet of Things (IoT) in recent years [1]–[7]. Following Microsoft's introduction of the CryptoNets [8] model for secure CNN prediction using FHE in 2016, the research community has explored the use of FHE in secure deep learning. However, many of these studies have primarily focused on algorithms such as CKKS [9] and BFV [10], where improvements to the neural network model structure were made to align with these algorithms. While these approaches incorporate single instruction multiple data (SIMD) techniques to enhance throughput, they are not suitable for providing a single inference service to an individual user. Microsoft's subsequent proposal, the Lola model [11], suffers from lower efficiency in ciphertext prediction and fails to meet the real-time requirement in typical cloud-based IoT applications.

This study aims to enable efficient ciphertext prediction for a single user in a cloud environment while ensuring that the response time of the entire interaction remains within the real-time requirement of less than 1 second. Furthermore, the method employed in this study maintains a single input and output interaction throughout the process while evaluating Convolutional Neural Network on encrypted data. The proposed approach assures real-time requirements are met while maintaining a unique input-output interaction.

The main contributions of this paper are outlined below:

- We propose a Confused Modulo Projection based Fully Homomorphic Encryption algorithm(CMP-FHE), a standard Fully Homomorphic Encryption algorithm, which can support floating point operations.
- We design and implement a practical general-purpose FHE library based on the CMP-FHE algorithm, and experiments show that our library has a high efficiency than the most commonly used FHE libraries like SEAL, HElib, and OpenFHE.
- We propose a method that enables our library to accelerate blind computation via GPU to improve efficiency based on the independent nature of the modulo projections. Our experiments show that our library is 841.67 to 3056.25 times faster than the state-of-the-art library [12], which is also based on confused modulo Projection.
- We develop a new FHE-friendly real-time homomorphic CNN (RT-HCNN) to classify images of ciphertext MNIST datasets after encryption using CMP-FHE. We use an obfuscated image parallel prediction method to accelerate the prediction of each layer using GPUs. RT-HCNN achieves a high accuracy rate of 99.13% and can evaluate an image in just 79.5ms, and is approximately 58 times faster than Microsoft's Lola [11].

The remainder of the article is organized as follows: Section II reviews related work, while Section III presents an overview of the CMP-SWHE algorithm along with the fundamental concepts underlying its limited and convolutional neural network (CNN) implementations. In Section IV, we present the CMP-FHE Cryptosystem. In Section V, We present the FHE library we designed and conduct comparative experiments. In Section VI, We discuss the challenges of using the CMP-FHE algorithm and our library in deep learning and propose corresponding solutions. Additionally, we introduce the RT-HCNN architecture and present the results of our experiments with the MNIST dataset. Finally, in Section VII, we present our conclusion and outline future research directions.

## II. Related Work

With the continuous advancement of deep learning, the research on privacy-preserving neural network prediction has garnered significant attention recently. Several privacy-preserving technologies have been developed and applied in conjunction with deep learning, including federated learning that employs differential privacy [13]–[18], multi-party computing (MPC) [19]–[23], and homomorphic encryption, among others.

Related to our work, a homomorphic encryption scheme for neural networks was first proposed by Orlandi et al. [24]. The proposed scheme focuses on resolving the challenges associated with nonlinear activation functions by establishing an interactive protocol between the data and model owners. Despite its promising potential, the scheme suffered a significant delay, significantly limiting its practicality. Dowlin et al. made a significant breakthrough in the application of homomorphic encryption in deep neural networks by introducing the CryptoNets model [8], which employed the leveled FHE [25] technology and SIMD batch processing to support high-throughput calculation. However, this model had limitations when classifying a single image, and the client had to generate encryption parameters according to the model structure, which could compromise the model's privacy. Hesamifard et al. proposed the CryptoDL model [26], which approximated the commonly used activation function in CNNs using a low-degree polynomial function to complete the training of neural networks. This model used the generated model in the training stage to classify encrypted data and achieved improved prediction accuracy, but the number of DNN layers limited it.

Following them, Bourse et al. proposed the FHE-DiNN model [27], in which they utilized Binary Neural Networks to enable encrypted prediction by dispersing the weights and inputs of traditional CNNs into $\{-1, 1\}$ alongside the fast bootstrapping of the TFHE scheme proposed by Chilotti et al. [28] was exploited to double as an activation function for neurons. Although the model performed well on the MNIST dataset, its recognition rate could be better. Sanyal et al. [29] proposed the TAPAS system and Encrypted Prediction as a Service (EPAAS) based on BNN to predict FHE-encrypted data. The scheme was based on binary and sparse technologies, which realized the acceleration and parallelism of complex models and allowed service providers to update the model structure. However, its application scope was limited to BNN. Jiang et al. [30] introduced a novel approach, E2DM transformed matrix multiplication into a computation performed using FHE and applied this technique to evaluate a neural network on the MNIST dataset. In comparison to Dowlin

et al., [8], who encrypted only one pixel per ciphertext but evaluated large batches of images, Jiang et al. [30] explored the approach of encrypting an entire image within a single ciphertext. Their evaluation demonstrated promising performance, as they could evaluate 64 images in approximately 29 seconds. However, their overall amortized performance was relatively lower.

Badawi et al. [31] introduced a high-performance variant of the somewhat homomorphic encryption scheme called the FV scheme, specifically designed for efficient execution on GPUs using CUDA. They then proceeded to present the first-ever GPU-accelerated homomorphic convolutional neural network HCNN [32], built upon the framework of CryptoNets. Their implementation yielded a remarkable improvement in prediction time, which was 40.41 times faster than its non-accelerated counterpart. However, despite this impressive performance boost, the approach still needed to meet the real-time requirements.

Recently, Jang et al. [33] introduced a deep sequence model that utilizes matrix homomorphic encryption for conducting to infer encrypted data. To address the challenge of computational efficiency associated with matrix homomorphic encryption, the authors devised an efficient matrix multiplication method and a suitable encryption parameter selection technique. In several experimental evaluations, they successfully demonstrated the effectiveness of their proposed approach; however, it is worth noting that the achieved accuracy on the MNIST dataset was 94.2%.

Jin et al. [12] proposed the CMP-SWHE encryption algorithm, which utilizes confused modulo projection homomorphism. This encryption scheme was then applied to various traditional multimedia applications in the context of the IoT, such as foreground extraction. As IoT applications integrate with artificial intelligence technology, it is becoming increasingly apparent that CMP-SWHE may no longer be sufficient to meet the privacy protection requirements of neural networks.

## III. Preliminaries

### A. Fully Homomorphic Encryption

Homomorphic Encryption was first proposed by Rivest et al. [34] in 1978. It wasn't until 2009 that Gentry [35] proposed the first viable FHE solution. Currently, the only viable approach to designing fully homomorphic encryption schemes involves a simple two-step process. First, a somewhat homomorphic encryption (SWHE) scheme, in which ciphertext cannot be evaluated for arbitrary circuits, is designed to evaluate its decryption function. Then, bootstrapping [35] is performed, which involves decrypting a ciphertext using an encrypted copy of the secret key. Although this process has been modernized, the fundamental ideas and steps remain unchanged.

Given the considerable computational complexity associated with bootstrapping, many researchers now prefer the leveled FHE [25] scheme, which enables function evaluation up to a pre-defined multiplicative depth without necessitating bootstrapping.

In 2020, Jin et al. [12] proposed a somewhat homomorphic encryption algorithm based on a confused modulo projection theorem named CMP-SWHE [12]. And an algorithm library is designed. The results of some traditional experiments with multimedia technologies show that this library is very efficient. But using the CMP-SWHE algorithm library for deep neural networks has the following problems:

1) The CMP-SWHE algorithm is limited in that it only supports integer operations. However, neural networks commonly represent weights and inputs as floating-point numbers, making them unsuitable for performing convolution operations crucial for CNNs. As a result, the CMP-SWHE algorithm cannot efficiently execute these essential operations required by CNNs.
2) The CMP-SWHE algorithm is not a fully homomorphic encryption algorithm but a somewhat homomorphic one. As such, it is not designed to handle the complex multiplication circuit depth required by neural networks.
3) Due to the high computational cost of predicting ciphertext neural networks, this algorithm further reduces the computational efficiency, making it unsuitable for real-time applications.

Our proposed CMP-FHE IV algorithm solves the above problem, and it has better performance.

### B. Convolutional Neural Network

CNN is a widely used deep learning model inspired by the human visual system. It has found applications in various fields, such as image recognition, object detection, and speech recognition, delivering exceptional performance. CNN consists of convolution layers, pooling layers, and fully connected layers. The convolution layers perform feature extraction from input images, while the pooling layers downsample the feature maps. The fully connected layers are responsible for classification or regression tasks. In our subsequent models, we will describe and utilize these operations.

*1) Convolution Layer:* In CNN, each convolution layer usually includes multiple convolution kernels. The convolution layer uses convolution operation to extract image features, and each convolution kernel can extract a feature in the input image, which can be stacked into a feature map. The convolution operation includes two parameters: convolution kernel and convolution step size. As shown in Fig. 2, the convolution kernel slides on the image and performs element multiplication and summation operations to generate a feature map.

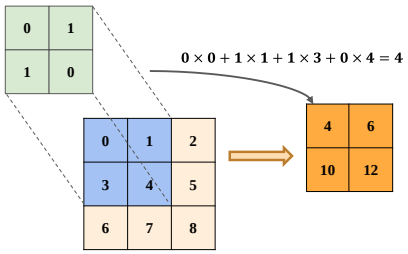


Fig. 2. Sliding Convolution.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3313443



*2) Pooling Layer:* The pooling layer is utilized to decrease the dimensionality of the feature map while preserving important features. Max pooling and average pooling are two commonly employed pooling operations. As shown in Fig. 3 and 4, they select the region's maximum value or average value as the pooled value, respectively.

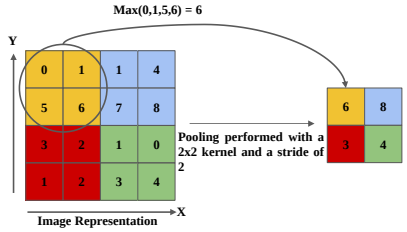


Fig. 3. Max Pooling.

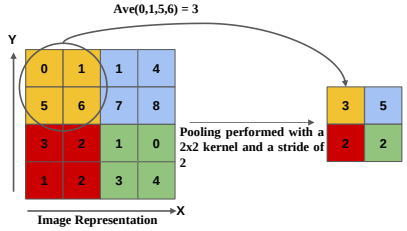


Fig. 4. Average Pooling.

*3) Activation Function:* As shown in Fig. 5, the activation function converts the output of neurons into a nonlinear form. The convolution layer and full connection layer usually use activation functions.


Fig. 5. Activation Function.

*4) Full Connection Layer:* As shown in Fig. 6, the fully connected layer treats the pixels within each feature map as individual neurons. It connects them to all neurons in the previous layer. By doing so, the fully connected layer can combine all the extracted features from the image and generate the entire image's final classification or regression results. It allows for integrating information from different parts of the image, enabling the network to make global predictions based on the learned features.

## IV. Confused Modulo Projection Based FHE Cryptosystem

In this section, we introduce the principles of the CMP-FHE algorithm for encoding floating point numbers and the five main parts of the algorithm.

### A. Floating-point Operations

One of the primary challenges we encountered was determining the appropriate plaintext space for HCNN computation. A floating-point number is the most common method to express real numbers in computers; the weights and inputs

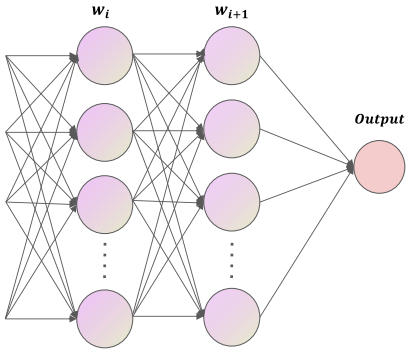


Fig. 6. Fully Connected.

of a neural network are typically represented in floating-point decimal format, which presents a potential issue. In this study, we adopt a scalar coding method, whereby floating-point numbers are converted to integer form by multiplying the explicit floating-point data by a scale factor $\Delta$ and rounding to the nearest integer. When decoding, we then divide out the scale factor. In the calculation, the ciphertext with the same scale factor can be directly calculated, while, when dealing with ciphertexts having different scale factors, We need to unify scale factors. For example, we have two plaintexts, $\Delta_1 m_1, \Delta_2 m_2$ and two different scale factors $\Delta_1 > \Delta_2$, $\frac{\Delta_1}{\Delta_2} = \Delta^l$. When we perform multiplication, it is easy to know:

$$\Delta_1 m_1 \times \Delta_2 m_2 = \Delta_1 \Delta_2 m_1 m_2$$
$$m_1 \times m_2 = \frac{\Delta_1 \Delta_2 (m_1 \times m_2)}{\Delta_1 \Delta_2}$$

while performing addition, due to the difference in scale factors, we cannot add directly, and we need to unify the scale factors in order to get the correct results when decoding. We need to do the following:

$$\Delta_2 m_2 \rightarrow \Delta_2 \Delta^l m_2$$
$$\Delta_1 m_1 + \Delta_2 \Delta^l m_2 = \Delta_1 (m_1 + m_2)$$
$$m_1 + m_2 = \frac{\Delta_1 (m_1 + m_2)}{\Delta_1}$$

### B. Algorithm

The CMP-FHE fully homomorphic algorithm consists of 5 parts: KeyGen, SecEncrypt, PubEncrypt, Blind Operation, and Decrypt. The first part generates the secret key(SK), the public key(PK) and the evaluation key(EK). The second part encodes the floating-point plaintext according to section IV-A, then encrypts by amplifying, randomizing, calculating modulo projections, and setting confusing redundant modulo projections. In the third part of public key encryption, the secret key encrypts the plaintext of 1 to derive the public key (PK). This operation is based on the property that multiplying any number by one yields the original value itself. Thus the ciphertext of 1 serves as the public key, and multiplying it with the plaintext gives its corresponding ciphertext. The fully homomorphic encryption algorithm can only perform addition and multiplication operations. In the fourth part, according to the Congruent equivalence property, we present four blind computational algorithms that include addition and multiplication. In the last section, we describe extracting the real modulo

projections and decrypting according to the Chinese remainder theorem(CRT) [36].

*1) KeyGen:* Our key parameters include the modulo bases $B$, magnification $a$, scale factor $\Delta$ and position template $S$. The original data is obtained as modulo projections by performing a modulo operation on $B$. The amplification factor $a$ is determined to eliminate noise and ensure the correctness of the result. The scale factor $\Delta$ is determined based on the adequate precision required for representing floating-point numbers. It ensures that the range and precision of the encrypted computations align with the desired accuracy of the results. The position template $S$ determines the position of each real modulo projection in the $M-1$ confusing redundant modulo projections.

The selection of the number of modulo bases $N$ and the number of confusing redundant modulo projections $M-1$ in CMP-FHE is influenced by the depth of the multiplication circuit. A minimum value of $N \log M > 128$ is required to ensure security. The term "128-bit security" in the security parameter refers to the level of security provided by a cryptographic algorithm using a 128-bit key length. In modern cryptography, this is considered to be sufficiently secure, and no effective attack has been found that can break a 128-bit key in a reasonable amount of time. Here, we require $N \log M > 128$ to ensure that an attacker cannot correctly guess the location of each real modulo projection by brute-force decryption. This ensures the security of the system.

In Algorithm 1, each user generates their secret key ($SK$), public key ($PK$), and evaluation key ($EK$). This algorithm outlines the steps in generating these keys, which are essential for encryption, decryption, and homomorphic computations.

**Algorithm 1** KeyGen

**Require:**
$U, N, M, a, \Delta$

**Ensure:**
$SK, PK, EK$

1: The initial step is to select the first $N$ numbers from the prime modulo base pool $B_n = \{b_1, b_2, \ldots, b_n\}$ based on the predefined number of modulo bases $N$ to create the modulo projection base $B$.
2: $U$ is encrypted using AES [37] in ECB mode to get position template $S$, where each $\log M$ bits correspond to an index $s_i$ representing the position of correct modulo projections. The indices are used during decryption to apply the appropriate modulo projections and obtain the accurate decrypted result, $0 \leq s_i < M-1$.
3: $PK$ is obtained by encrypting the plaintext of 1 with $SK$, and the secret key encryption algorithm is shown in Algorithm 2.
4: $SK$ consists of the modulo projection base $B$, the position template $S$, the magnification factor $a$, and the scale factor $\Delta$, $PK$ consists of ciphertext of 1, the modulo projection base $B$, the magnification factor $a$, and the scale factor $\Delta$ , while $EK$ consists of the modulo projection base $B$, the magnification factor $a$, and the scale factor $\Delta$.

*2) SecEncrypt:* Now, we use a secret key to encrypt the original data. Suppose the number of modulo bases is 64, and the number of confusing redundancies is 3. For each plaintext and redundant data, a random noise $\eta$ is added. The purpose is to prevent the same plaintexts and modulo bases will produce the same modulo projections and the same ciphertext, which needs to satisfy $\eta << a$, in order to make, according to equation 2, decryption eliminates the random number and its derived noise and get the correct result. The original data modulo each corresponding modulo base will get 64 real modulo projections, each real modulo projection is mixed with 3 confusing redundancies, and there are four optional positions for each real projection, and the location of the real modulo projection at each position is determined by the position template $S$ in the secret key($SK$). A ciphertext of size $64 \times 4$ will be obtained in the end. The magnification $a$, like the scale factor $\Delta$ IV-A, also needs to be unified for the magnification when performing the addition, and we use $l$ and $t$ in the ciphertext to record the change in magnification $a$ and scale factor $\Delta$ during the calculation.

Algorithm 2 describes the process of encrypting a known plaintext $P$ to ciphertext $e$ with a secret key, and the process of generating a public key requires $P = 1$.

*3) PubEncrypt:* Since FHE itself has the characteristics of asymmetric cipher and $PK$ is the result of secret key encryption of plaintext of 1, the process of public key encryption is completed by multiplying $PK$ with plaintext once, Algorithm 3 describes the specific process.

*4) Blind Operation:* The homomorphism principle of the CMP-FHE algorithm is based on the Congruent equivalence property and its extension theorem, as shown in equation 1.

$$\begin{aligned}(x+y) \bmod b &\equiv ((x \bmod b) + (y \bmod b)) \bmod b \\ (x \times y) \bmod b &\equiv ((x \bmod b) \times (y \bmod b)) \bmod b\end{aligned} \tag{1}$$

As can be seen from equation 1, we equate the corresponding addition or multiplication computation for each plaintext modulo projection to the same addition and multiplication operations for the plaintext, satisfying the homomorphism requirement.

Here we have four blind computation operations addition of ciphertext and ciphertext $BlindAdd$, multiplication of ciphertext and ciphertext $BlindMul$, the addition of ciphertext and plaintext $HalfBlindAdd$, multiplication of ciphertext and plaintext $HalfBlindMul$, we assume that there are two ciphertexts $C_1, C_2$, and plaintext floating point number $K$, the order of $C_1$ magnification is $t_1$, the order of the scale factor of $C_1$ is $l_1$, the order of $C_2$ magnification is $t_2$, the order of the scale factor of $C_2$ is $l_2$, and $t_1 < t_2, l_1 > l_2$. The order of the amplification of the resultant ciphertext $C$ is $t$, and the order of the scale factor is $l$. Algorithm 4 presents the specific procedure for performing the above four blind calculations using $C_1$ and $C_2$ according to equation 1 and section IV-A.

*5) Decrypt:* For decryption, we take out the correct modulo projections from the ciphertext result based on the position template $S$ in the $SK$ and use CRT [36] to obtain the computation result of the original data. As the multiplication circuit becomes more profound, the noise level in the system

**Algorithm 2** SecEncrypt

**Require:**
$SK, P$

**Ensure:**
Secret key encrypted ciphertext $e$

1: $R = \{R_1, R_2, \ldots, R_M\}$ which is a random integer array that is generated for obfuscating the real modulo projections.

2: Multiply the floating point number by the scaling factor $\Delta$, rounded to the nearest integer, and scale up with $a$ and randomize plaintext $P$, $a >> \eta$.

$$P^{'} = round(\Delta \times P) \times a + \eta$$

3: Amplify and randomize redundant data $R$, $a >> \eta$.

$$R_i{}' = aR_i + \eta, R' = \left\{R_1{}', R_2{}', \ldots, R_{M-1}{}'\right\}$$
$$i = 1, 2, \cdots, M$$

4: Compute the modulo projections for $P^{'}$ to obtain the correct modulo projections $p$.

$$p = \{p_1, p_2, \ldots, p_N\}, p_i = P' \mod b_i$$
$$i = 1, 2, \cdots, N$$

5: Compute the modulo projections for $R^{'}$ to obtain the confusing modulo projections $r$.

$$r = \{r_1, r_2, \ldots, r_N\}$$
$$r_i = \{r_{i_1}, r_{i_2}, \ldots, r_{i_M}\}, i = 1, 2, \ldots, N$$
$$r_{i_j} = R_i{}' \mod b_i, j = 1, 2, \ldots, M$$

6: By utilizing the position template $S$, the correct modulo projections $p$ are inserted into the array of confusing redundant modulo projections $r$, resulting in a ciphertext $e$ of size $N \times M$.

$$e = \text{SecE}ncrypt(P) = \{e_1, e_2, \ldots, e_N\}$$
$$e_i = \{e_{i_1}, e_{i_2}, \ldots, e_{i_M}\}, i = 1, 2, \ldots, N$$
$$e_{i_j} = \begin{cases} r_{i_j}, & j \neq s_i \\ p_i, & j = s_i \end{cases}$$
$$i = 1, 2, \ldots, N, j = 1, 2, \ldots, M$$

7: Initialize the order $t$ of the magnification $a$ and the order $l$ of the scale factor $\Delta$.

$$t = 1, l = 1$$

8: Ciphertext $e$ includes $l$ and $t$.

**Algorithm 3** PubEncrypt

**Require:**
$PK, P$

**Ensure:**
Public key encrypted ciphertext $c$

1: Convert plaintext $P$ to integer.

$$P^{'} = round(\Delta \times P)$$

2: Multiply the modulo projection in $PK$ with $P^{'}$.

$$c_{i_j} = (pk_{i_j} \times P^{'}) \bmod b_i$$
$$i = 1, 2, \ldots N, j = 1, 2 \ldots M$$

3: Update the order $t$ of magnification $a$ and the order $l$ of scale factor $\Delta$ in $c$.

$$t = 1, l = 2$$

## V. CMP-FHE Parallel Library

To simplify the use of the CMP-FHE algorithm for users and enable them to apply it directly in various application scenarios without needing to understand the specific details of the algorithm, we have integrated it into a general algorithm library based on the CUDA C++ language. By providing this integration, users are able to make use of the functionality provided by the CMP-FHE algorithm without requiring them to have specialized knowledge or skills.

### A. GPUs accelerated blind computing

GPUs consist of hundreds of cores, each capable of processing different data, making them more suitable for large-scale data-parallel computing tasks than CPUs. Thanks to the inherent properties of the CMP-FHE algorithm, the operations between different modulo projections are independent, which makes it well-suited for GPUs acceleration. As depicted in Fig.7, the CPU still handles the encryption and decryption process, while the blind computation process is offloaded to the GPUs for faster processing.

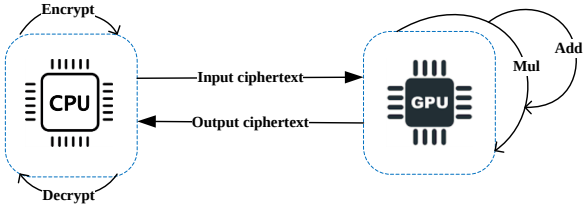


Fig. 7. Algorithm flow under GPU acceleration.

### B. Library Performance

We have compared our library with other libraries. The details of our experimental setup are provided in Table I.

*1) Comparison with FHE library:* We have compared the speed of floating point addition and multiplication with some of the leading FHE libraries, including HElib, SEAL, and OpenFHE, and the results are shown in the Table II.

increases. Theoretically, we can increase the number of modulo bases and enlarge the amplification factor $a$. According to the requirement of the unique solution of the CRT [36], $w*\Delta^l(a*\max(P))^t < \prod_{i=1}^{i=N} b_i$, where $w$ represents the number of terms in the polynomial coefficient, $\Delta$ is the scaling factor, $l$ is the order of the scaling factor, $a$ is the amplification factor, $P$ is the floating-point number plaintext, and $t$ is the order of the amplification factor. The specific process is described in Algorithm 5.

**Algorithm 4** Blind Operation

**Require:**
$EK, C_1, C_2, K$

**Ensure:**
Result of $C$

1: $t$ is the order of the magnification of $C$ and $l$ is the order of the C scale factor, $t_1$ is the order of the magnification of $C_1$ and $l_1$ is the order of the $C_1$ scale factor, $t_2$ is the order of the magnification of $C_2$ and $l_2$ is the order of the $C_2$ scale factor.

2: Calculate $BlindAdd(EK, C_1, C_2)$, homomorphic addition of two ciphertexts.

3: Align the order of magnification $a$ and scaling factor $\Delta$, sum of corresponding modulo projections.

$$c_{i_j} = (c_{1_{i_j}} \times a^{t_2 - t_1} + c_{2_{i_j}} \times \Delta^{l_1 - l_2}) \bmod b_i$$
$$i = 1, ...N, j = 1, ...M$$

4: Update the order $t$ of magnification $a$ and the order $l$ of scale factor $\Delta$.

$$t = t_2, l = l_1$$

5: Calculate $BlindMul(EK, C_1, C_2)$, homomorphic multiplication of two ciphertexts.

6: Multiply corresponding modulo projections.

$$c_{i_j} = (c_{1_{i_j}} \times c_{2_{i_j}}) \bmod b_i$$
$$i = 1, ...N, j = 1, ...M$$

7: Update the order $t$ of magnification $a$ and the order $l$ of scale factor $\Delta$.

$$t = t_1 + t_2, l = l_1 + l_2$$

8: Calculate $HalfBlindAdd(EK, C_1, K)$, homomorphic addition of a ciphertext and a floating-point plaintext.

9: Convert plaintext $K$ to integer.

$$K^{'} = round(\Delta \times K)$$

10: Align the order of magnification $a$ and scaling factor $\Delta$, all modulo projections sum the plaintext integer.

$$c_{i_j} = (c_{1_{i_j}} + K^{'} \times a^{t_1} \times \Delta^{l_1 - 1}) \bmod b_i$$
$$i = 1, 2, ...N, j = 1, 2, ...M$$

11: Update the order $t$ of magnification $a$ and and the order $l$ of scale factor $\Delta$.

$$t = t_1, l = l_1$$

12: Calculate $HalfBlindMul(EK, C_1, K)$, homomorphic multiplication of a ciphertext and a floating-point plaintexts.

13: Convert plaintext $K$ to integer.

$$K^{'} = round(\Delta \times K)$$

14: All modulo projections are multiplied by the plaintext integer.

$$c_{i_j} = (c_{1_{i_j}} \times K^{'}) \bmod b_i$$
$$i = 1, 2, ...N, j = 1, 2, ...M$$

15: Update the order $t$ of magnification $a$ and and the order $l$ of scale factor $\Delta$.

$$t = t_1, l = l_1 + 1$$

**Algorithm 5** Decrypt

**Require:**
$SK, f(C)$

**Ensure:**
Result of $f(P)$

1: Applying the position template $S$ in $SK$, the correct modulo projections of the calculation results $p = \{p_1, p_2, \cdots, p_N\}$ are extracted.

2: Decrypt the results of $f(P)$ according to CRT [36].

$$f(P) = \frac{\left(\sum_{i=1}^{i=N} d_i B_i B_i^{-1}\right) \bmod B_s}{a^t \Delta^l}, i = 1, 2, \ldots, N \quad (2)$$

3: Equation 2 defines the variables $t$ and $l$ as follows: $t$ represents the highest number of magnification $a$ present in the resulting polynomial, while $l$ corresponds to the order of scale factor $\Delta$, $B_s = \prod_{i=1}^{i=N} b_i$, $B_i = B_s / b_i$, $B_i B_i^{-1} = 1 \bmod b_i$.

TABLE I
CONFIGURATION OF THE TESTBED SERVERS

| Item | Specification | Item | Specification |
|---|---|---|---|
| CPU | Intel(R)Xeon(R)Gold6258R | GPU | NVIDIA GeForce RTX 3090 |
| #of Cores | 28 | #of Cores | 10496 |
| #of Threads | 112 | Compute Capability | 8.6 |
| CPU Frequency | 2.70GHz | GPU Core Frequency | 1695 MHz |
| System Memory | 24GB×4 DDR4 | GPU Core Frequency | 24GB |
| OS | Ubuntu 20.04.1 | CUDA Version | 10.2 |

*2) Comparison with SWHE library:* For a homomorphic library that is also based on modulo projection, we compared the speedup with Jin et al. [12] for doing 10,000 operations. We took 100, 200, and 300 moduli bases, 6-bit positions of the confusing modulo projection. We tested the GPU speedup for blind addition and blind multiplication with different modulo bases. Table III gives the experimental results. As the number of moduli increases, the computation time of Jin et al.'s blind computation shows a linear increase. In contrast, the GPU's computation time remains stable, and the blind computation speedup using the GPU is substantially improved.

*3) Comparison of traditional multimedia applications:* We also performed the blind foreground extraction algorithm using

TABLE II
TIME COMPARISON ON 3 FHE LIBRARIES

| Library | SEAL(ms) | HElib(ms) | OpenFHE(ms) | Ours(ms) |
|---|---|---|---|---|
| Encrypt | 6.22 | 3.95 | 60.23 | 0.22 |
| BlindAdd | 0.17 | 1.07 | 17.52 | 0.054 |
| BlindMul | 6.43 | 56.69 | 182.74 | 0.047 |
| Decrypt | 4.88 | 41.44 | 35.53 | 0.508 |

TABLE III
RUNTIME AND ACCELERATION RATIO OF BLIND COMPUTING ON GPU AND CPU

| Blind Operation | Modulo Bases | jin et al. [12] (s) | Ours(s) | Speedup |
|---|---|---|---|---|
| BlindAdd | 100 | 2.02 | 0.0024 | 841.67× |
| | 200 | 4.13 | 0.0025 | 1652× |
| | 300 | 6.14 | 0.0025 | 2456× |
| BlindMul | 100 | 1.63 | 0.0016 | 1018.75× |
| | 200 | 3.27 | 0.0016 | 2043.75× |
| | 300 | 4.89 | 0.0016 | 3056.25× |

TABLE IV
COMPARISON OF GPU-ACCELERATED CMP-FHE ALGORITHM LIBRARY AND CMP-SWHE ALGORITHM LIBRARY IN TERMS OF SPEED OF FOREGROUND EXTRACTION

| Algorithm | Image size | Library | Times(s) |
|---|---|---|---|
| Background-Difference | $720 \times 576$ | jin et al. [12] | 16.05 |
| | | Ours | 0.000006175 |
| Frame-Difference algorithm | $640 \times 480$ | jin et al. [12] | 11.88 |
| | | Ours | 0.000006175 |

the GPU-accelerated algorithm and compared the experimental results with those of Jin et al. [12], (a) is the background frame, the result denoted as (c) is obtained through blind foreground extraction applied to the input denoted as (b) using the Background-Difference algorithm. The frames labeled (d) and (e) represent two consecutive video frames. Frame (e) is obtained by applying blind foreground extraction to frame (d) using the Frame-Difference algorithm. With 20 modulo bases and 64 modulo projection positions, we can perform parallel blind foreground extraction on all pixel values of the ciphertext image simultaneously by sending them all to the GPU at once. This contrasts the approach Jin et al. [12] used, which computed the ciphertext pixel values one by one. The experimental results are shown in Fig. 8, as shown in Table IV, which shows that GPU acceleration has dramatically improved the speed of our library.

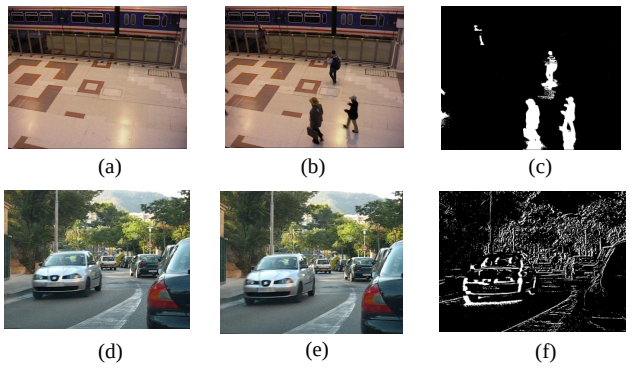


Fig. 8. Blind foreground extraction under GPU acceleration.

## VI. APPLICATIONS ON NEURAL NETWORKS: RT-HCNN

Since the FHE algorithm typically supports only a finite number of addition and multiplication operations, it can be used for simple polynomial functions. Therefore, modifications are required in the traditional CNN network structure to accommodate the fully homomorphic encryption mode.

### A. Description of the Network

*1) Convolution and Fully Connected Layer:* Since the convolutional and fully-connected layers are weighted sums, computing them over the encrypted data is straightforward. The weights can be multiplied with the encrypted input using HalfBlindMul, and the result added with the bias using HalfBlindAdd as shown in Algorithm 4.

*2) Approximating Non-Polynomial:* The activation function used in the product neural network is often nonlinear. Since homomorphic operations can only perform addition and multiplication, we cannot directly compute these activation functions. Indeed, in order to approximate the activation function accurately, it is essential to utilize polynomials. The choice of activation function often depends on the expected performance of the network. We use interpolation, least squares, Chebyshev, and $x^2$ to approximate Relu, and the accuracy obtained is shown in Table V. After conducting experiments, we decided to use $x^2$ as an approximation of the ReLU function. The validity of this polynomial substitution has been well-tested in models such as CryptoNets. Using $x^2$ as a low-degree polynomial can reduce the number of multiplications required, thereby ensuring the efficiency of the network under ciphertext.

TABLE V
PREDICTION ACCURACY OF MNIST PLAINTEXT MODEL WITH DIFFERENT APPROXIMATION METHODS

| Method | Polynomial order | Accuracy(%) |
|---|---|---|
| Interpolation | 2 | 97.86 |
| Least squares | 2 | 99.02 |
| Chebyshev | 2 | 98.79 |
| $x^2$ | 2 | 99.4 |

*3) Handling Pooling Layers:* In homomorphic operation, comparison operations are not feasible for activation functions. Therefore, an average set layer is utilized in this context, which does not require division but instead multiplies by the floating-point number corresponding to the divisor. For example, when the pooling layer window size is $2\times2$, $avg(x) = 0.25\times\sum_{i=1}^{n} x_i$.

Table VI presents the neural network architecture used in our experiments on the MNIST dataset. Both the training and testing phases utilize this network structure.

TABLE VI
THE RT-HCNN ARCHITECTURE IS EMPLOYED FOR TRAINING AND TESTING THE MNIST DATASET, INCORPORATING THE SCALE FACTOR UTILIZED IN SCALAR ENCODING

| Layer Type | Description | Layer Size |
|---|---|---|
| Convolution | Convolution kernel size $6 \times 5 \times 5$ and stride $(1, 1)$ with padding 0. | $28 \times 28 \times 6$ |
| Square | Previous layer outputs squared for next layer inputs. | $28 \times 28 \times 6$ |
| Pooling | Average pooling with extent 2 and stride 2. | $14 \times 14 \times 6$ |
| Convolution | Convolution kernel size $16 \times 5 \times 5$ and stride $(1, 1)$ without padding. | $10 \times 10 \times 16$ |
| Square | Previous layer outputs squared for next layer inputs. | $10 \times 10 \times 16$ |
| Pooling | Average pooling with extent 2 and stride 2. | $5 \times 5 \times 16$ |
| Convolution | Convolution kernel size $120 \times 5 \times 5$ and stride $(1, 1)$ without padding. | $1 \times 1 \times 120$ |
| Square | Previous layer outputs squared for next layer inputs. | $1 \times 1 \times 120$ |
| Fully Connected | Weighted sum that generates 10 outputs. | $1 \times 1 \times 10$ |

### B. Homomorphic networks with GPU-accelerated inference

Instead of parallelizing modulo projections within a single-blind computation operation, we achieve parallelism across the entire inference process. To do so, we create a new image composed of modulo projection at the exact location, enabling parallel inference across all new confused images. This approach dramatically accelerates the overall inference process.

In Fig. 9, we see an example of parallel prediction under ciphertext using the CMP-FHE algorithm library. Consider an

image of size $3\times3$, and we want to perform parallel prediction using ciphertext. We have 2 modulo bases $B = \{b[0], b[1]\}$, each having 1 confusion position and 1 true position. Each pixel value after encryption has four modulo projections numbers 0-3. Before prediction, we decompose the original image's encrypted image and combine the modulo projections with the same position of the same modulo base into 4 new ciphertext confused sub-images. Then, the 4 ciphertext confusion images are predicted in parallel using the GPU. Finally, when decrypting, we extract the predicted results of the 2 correct modulo projections and decrypt them to obtain the final prediction result. This parallelization method accelerates the whole inference process by achieving parallelism of the whole inference process instead of parallelizing the modulo projections inside a single-blind computation operation.

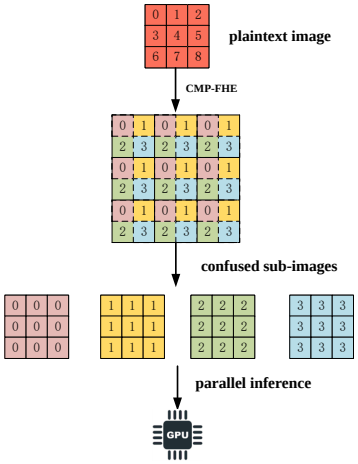


Fig. 9. Ciphertext image decomposition.

### C. Experiments and Comparisons

This section presents the results of our experiments on classification under ciphertexts, which consist of two parts. First, a neural network is trained on the plaintext to obtain the model's parameters, and then the data is encrypted using CMP-FHE and classified under ciphertexts.

The MNIST dataset [36] has served as a prevalent benchmark for classifiers. Our experiments utilized the MNIST dataset, which comprises 60,000 black-and-white images of handwritten digits ranging from 0 to 9. The dataset consists of 50,000 images for training and 10,000 images for prediction. Each image is represented by grayscale values ranging from 0 to 255. Pytorch was used to train a 9-layer neural network shown in Table VI on the MNIST dataset, which achieved an accuracy of 99.4%, as shown in Table V.

### D. Choice of Parameters

To align the output of each layer of the ciphertext model with the plaintext model, we adjust the number of modulo bases. Fig. 10 shows the size of the modulo bases that we tested to guarantee the correct results for each layer under the depth network of Table VI, with the magnification $a = 1000000$, according to the accuracy of the retained plaintext floating point numbers, the scale factor $\Delta = 1000000$. Based on the experimental results, it is clear that the number of modulo bases can be adjusted to achieve leveled FHE. Furthermore, it is observed that the number of modulo bases approximately doubles after passing through the 2nd, fifth, and eighth square layers. Hence, it can be concluded that the number of activation layers and the degree of the polynomial significantly influence the count of modulo bases.

In order to fully utilize the available resources, it is preferable to have the number of threads be a multiple of the CUDA thread Warp, which is 32. Therefore, we select 96 modulo bases, which is a multiple of the thread Warp and three redundant modulo projections, which can ensure a security level of 192 bits. The parameters of the experimental environment are shown in Table I, and the parameters of the CMP-FHE algorithm are shown in Table VII.

TABLE VII
HE PARAMETERS FOR MNIST

| RT-HCNN | Modulo Bases | confusing redundancies | $a$ | $\Delta$ | Estimated security |
|---|---|---|---|---|---|
| MNIST | 96 | 3 | 1000000 | 1000000 | 192bits |

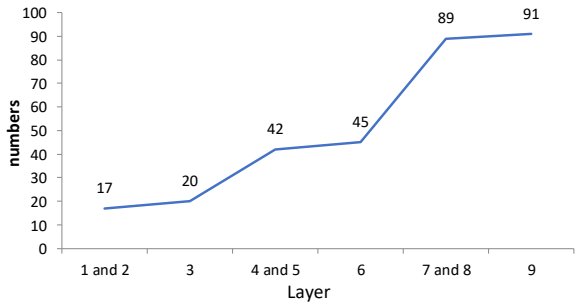


Fig. 10. Minimum number of modulo bases required for each layer.

### E. RT-HCNN Performance

Using the CMP-FHE algorithm for encryption, we measure the time it takes to predict a single instance using our proposed method. Fig. 11 shows the running time of each layer of RT-HCNN under CPU. It can be observed that the convolution and square layers consume the most time, and is far from real-time, so we need to perform accelerated inference.

To speed up the prediction process, we pre-calculate the parameters that align the magnification and scaling factors during the prediction process. In fact, for a specific network and alignment process, the CMP-FHE algorithm has fixed parameters, which are different from those of the plaintext network calculation process, in addition to weight and bias parameters. This is one of the reasons why the CMP-FHE algorithm is efficient. Moreover, the order of the final encrypted magnification and scaling factors is also determined, so there is no need to recalculate the order of the ciphertext

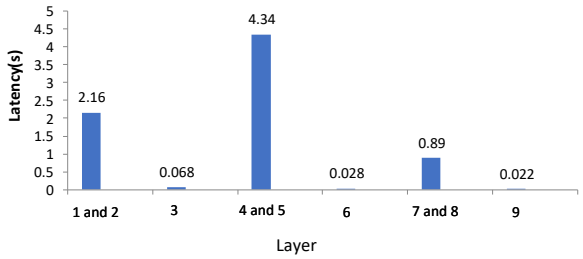


Fig. 11. Breakdown of time required to apply to MNIST network under CPU.

TABLE VIII
ACCURACY COMPARISON: PRIOR FHE-BASED PRIVACY-PRESERVING NEURAL NETWORKS VS. RT-HCNN

| Model | Layer number | DL technique | Accuracy(%) |
|---|---|---|---|
| CryptoNets [8] | 5 | CNN | 98.85 |
| FHE-DiNN100 [27] | 5 | BNN | 96.35 |
| LoLa [11] | 6 | CNN | 98.95 |
| HCNN [32] | 5 | CNN | 99 |
| Jang et al. [33] | - | RNN | 94.2 |
| RT-HCNN | 9 | CNN | 99.13 |

during the encryption inference process, further simplifying the encryption inference process. It also dramatically reduces the size of the transmitted ciphertext.

The bandwidth consumption of the CMP-FHE fully homomorphic encryption algorithm primarily depends on the number of modulo bases. In the case of the MNIST network described in the paper, with a parameter setting of 96 modulo bases and three redundant modulo projections, the ciphertext is sent as multiple obfuscated images. Each obfuscated image has a size of $32 \times 32 \times 2$ bytes, which is equivalent to 2 KB. There are a total of 384 obfuscated images ($96 \times 4$). The returned ciphertext has a size of $96 \times 4 \times 2 \times 10$ bytes, which amounts to approximately 7.5KB. According to the Chinese Residual Theorem, as the multiplication depth increases, the larger the resultant ciphertext is, the more modulo bases are required, and the higher the bandwidth usage.

To test the ciphertext prediction speed under GPU acceleration, we first decompose the image according to the method of VI-B and then use CUDA's nvprof command to test the average time of allocating a thread to predict a confusing image under GPU is 61.4ms, and then we test entirely the time of inferring the whole images, where encryption and decryption are performed under CPU, and prediction is performed under GPU.

Table VIII compares the prediction accuracy and evaluation time between prior FHE-based privacy-preserving neural networks and RT-HCNN. Furthermore, Table IX comprehensively compares the prediction times for prior privacy-preserving neural networks and RT-HCNN in the context of single-user prediction for a single sample.

TABLE IX
COMPARISON OF PREDICTION PROCESS TIME: PRIOR FHE-BASED PRIVACY-PRESERVING NEURAL NETWORKS VS. RT-HCNN

| Model | Time enc(s) | Time eval(s) | Time dec(s) |
|---|---|---|---|
| FHE-DiNN100 [27] | 0.000168 | 1.65 | 0.0000106 |
| LoLa [11] | 0.192 | 4.61 | 0.2029 |
| SHE [38] | - | 9.3 | - |
| HCNN [32] | - | 14.1 | - |
| RT-HCNN | 0.03003 | 0.0795 | 0.0203 |

### F. Analysis

Here the accuracy under plaintext is 99.4%, and the model accuracy under ciphertext is 99.13%. The main reason is that the Pytorch model parameters saved down the plaintext parameters must be more accurate. From the model parameters results, the four decimal places of reasonable accuracy are retained; if we can achieve fixed accuracy training, then theoretically, we can use CMP-FHE homomorphic algorithm library to achieve accuracy under ciphertext equal to that under plaintext. As seen from VI-E, the time taken for parallel prediction of multiple confusion images is 79.5ms, which is very close to the time to predict a single image at 61.4ms under GPU. When the prediction accuracy of our model is comparable to other homomorphic neural network models, the prediction time meets the real-time application requirements. The successful implementation of the CMP-FHE algorithm library on this more complex network structure demonstrates its potential for use in more advanced machine learning models.

### G. IoT Applications Prospects

Our algorithm holds a significant advantage over other lightweight models in IoT applications. The real-time capabilities of our homomorphic algorithm make it suitable for a wide range of application scenarios.

In intelligent transportation, neural networks can analyze traffic flow, predict traffic congestion, and provide real-time route planning and navigation. Protecting the location and trip privacy of vehicle owners and drivers while utilizing these networks is essential.

Real-time monitoring and security using cameras and vision sensors are also crucial. Neural networks can perform real-time analysis of video streams on edge devices, enabling the detection of abnormal behavior and recognizing faces. Privacy-preserving techniques can be employed to anonymize or de-identify sensitive information.

In the medical field, real-time monitoring of medical information, such as physiological parameters and medication intake, is possible through IoT devices and sensors. Neural networks can analyze this data and provide telemedicine diagnoses and recommendations while ensuring patients' medical information privacy.

By integrating with edge computing, we can encrypt the model parameters and utilize the ciphertext model for data computation on the edge device. Subsequently, the computed predictions are sent back to the client in ciphertext form,

which are then decrypted to obtain the desired results.For instance, in the scenario of identifying a specific person in a surveillance video, the end device inputs the plaintext video data to the ciphertext model for computation. The resulting ciphertext prediction is then sent back to the client without the need to transmit the entire video continuously. This approach significantly reduces the amount of data to be transmitted. Moreover, by using a ciphertext model on the end device, even if the device is lost, there is no risk of exposing the model data, thus providing an additional layer of security for the device. Through the combination of edge computing and homomorphic encryption, our algorithm not only enhances efficiency but also ensures security. This methodology optimizes data transmission and processing, thereby improving overall efficiency while preserving data privacy and model security.

These applications rely on real-time shared intelligence while also demanding privacy protection. Undoubtedly, the real-time performance of the homomorphic algorithm will play a crucial role. By adjusting the number of modulo bases and incorporating other techniques, we can expand the depth of the network. Additionally, we can customize the network model implementation to align with the specific privacy protection requirements.

## VII. Conclusion and Future Work

This paper proposes an efficient, fully homomorphic algorithm CMP-FHE that supports floating-point operations and provides the corresponding library. The application of our library to privacy-preserving neural networks for secure ciphertext image classification is demonstrated. An RT-HCNN network for MNIST handwriting recognition is designed and implemented, achieving an accuracy of 99.13% under a leveled FHE scheme with 192-bit secure ciphertexts. The whole inference process is executed in less than 1s using parallel inference with GPUs, meeting the real-time requirement and demonstrating the excellent efficiency advantage of the CMP-FHE algorithm.

Experimental results indicate that the bottleneck affecting the CMP-FHE algorithm is expanding the number of modulo bases, and further optimization work can be pursued. Our future endeavors will prioritize investigating this technology's capabilities in building scalable, distributed, and optimized artificial neural networks for IoT. We aim to explore novel approaches that harness the potential of cloud computing and parallel processing and ensure privacy protection measures to develop advanced IoT neural network architectures.

## Acknowledgment

This work is partially supported by Fundamental Research Funds for the Central Universities(328202242). The work of K.-K. R. Choo was supported only by the Cloud Technology Endowed Professorship. For the reproducibility of the proposed method, we have published our source code online at https://github.com/BESTICSP/RT-HCNN.

## References

[1] L. Zhang, J. Xu, P. Vijayakumar, P. K. Sharma, and U. Ghosh, "Homomorphic encryption-based privacy-preserving federated learning in iot-enabled healthcare system," *IEEE Transactions on Network Science and Engineering*, 2022.

[2] Y. Wang, X. Liang, X. Hei, W. Ji, and L. Zhu, "Deep learning data privacy protection based on homomorphic encryption in aiot," *Mobile Information Systems*, vol. 2021, pp. 1–11, 2021.

[3] V. Subramaniyaswamy, V. Jagadeeswari, V. Indragandhi, R. H. Jhaveri, V. Vijayakumar, K. Kotecha, and L. Ravi, "Somewhat homomorphic encryption: Ring learning with error algorithm for faster encryption of iot sensor signal-based edge devices," *Security and Communication Networks*, vol. 2022, 2022.

[4] R. Praveen and P. Pabitha, "Improved gentry–halevi's fully homomorphic encryption-based lightweight privacy preserving scheme for securing medical internet of things," *Transactions on Emerging Telecommunications Technologies*, p. e4732, 2023.

[5] H. S. Trivedi and S. J. Patel, "Homomorphic cryptosystem-based secure data processing model for edge-assisted iot healthcare systems," *Internet of Things*, p. 100693, 2023.

[6] D. Huang, Q. Gan, X. Wang, M. R. Ogiela, and X. A. Wang, "Privacy preserving iot-based crowd-sensing network with comparable homomorphic encryption and its application in combating covid19," *Internet of Things*, vol. 20, p. 100625, 2022.

[7] J. L. López Delgado, J. A. Álvarez Bermejo, and J. A. López Ramos, "Homomorphic asymmetric encryption applied to the analysis of iot communications," *Sensors*, vol. 22, no. 20, p. 8022, 2022.

[8] 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 *International conference on machine learning*. PMLR, 2016, pp. 201–210.

[9] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer, 2017, pp. 409–437.

[10] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. Springer, 2012, pp. 868–886.

[11] A. Brutzkus, R. Gilad-Bachrach, and O. Elisha, "Low latency privacy preserving inference," in *International Conference on Machine Learning*. PMLR, 2019, pp. 812–821.

[12] X. Jin, H. Zhang, X. Li, H. Yu, B. Liu, S. Xie, A. K. Singh, and Y. Li, "Confused-modulo-projection-based somewhat homomorphic encryption—cryptosystem, library, and applications on secure smart cities," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6324–6336, 2020.

[13] J. Park and H. Lim, "Privacy-preserving federated learning using homomorphic encryption," *Applied Sciences*, vol. 12, no. 2, p. 734, 2022.

[14] J. Ma, S.-A. Naas, S. Sigg, and X. Lyu, "Privacy-preserving federated learning based on multi-key homomorphic encryption," *International Journal of Intelligent Systems*, vol. 37, no. 9, pp. 5880–5901, 2022.

[15] Z. Shi, Z. Yang, A. Hassan, F. Li, and X. Ding, "A privacy preserving federated learning scheme using homomorphic encryption and secret sharing," *Telecommunication Systems*, vol. 82, no. 3, pp. 419–433, 2023.

[16] W. Jin, Y. Yao, S. Han, C. Joe-Wong, S. Ravi, S. Avestimehr, and C. He, "Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system," *arXiv preprint arXiv:2303.10837*, 2023.

[17] H. Ku, W. Susilo, Y. Zhang, W. Liu, and M. Zhang, "Privacy-preserving federated learning in medical diagnosis with homomorphic re-encryption," *Computer Standards & Interfaces*, vol. 80, p. 103583, 2022.

[18] R. Wang, J. Lai, Z. Zhang, X. Li, P. Vijayakumar, and M. Karuppiah, "Privacy-preserving federated learning for internet of medical things under edge computing," *IEEE Journal of Biomedical and Health Informatics*, 2022.

[19] O.-A. Kwabena, Z. Qin, T. Zhuang, and Z. Qin, "Mscryptonet: Multi-scheme privacy-preserving deep learning in cloud computing," *IEEE Access*, vol. 7, pp. 29 344–29 354, 2019.

[20] M. Li, S. S. Chow, S. Hu, Y. Yan, C. Shen, and Q. Wang, "Optimizing privacy-preserving outsourced convolutional neural network predictions," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1592–1604, 2020.

[21] M. Dahl, J. Mancuso, Y. Dupis, B. Decoste, M. Giraud, I. Livingstone, J. Patriquin, and G. Uhma, "Private machine learning in tensorflow using secure computation," *arXiv preprint arXiv:1810.08130*, 2018.

[22] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 19–38.

[23] V. Chen, V. Pastro, and M. Raykova, "Secure computation for machine learning with spdz," *arXiv preprint arXiv:1901.00329*, 2019.

[24] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing via homomorphic encryption," *EURASIP Journal on Information Security*, vol. 2007, pp. 1–11, 2007.

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

[26] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv:1711.05189*, 2017.

[27] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part III 38*. Springer, 2018, pp. 483–512.

[28] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.

[29] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade, "Tapas: Tricks to accelerate (encrypted) prediction as a service," in *International conference on machine learning*. PMLR, 2018, pp. 4490–4499.

[30] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1209–1222.

[31] A. Al Badawi, B. Veeravalli, C. F. Mun, and K. M. M. Aung, "High-performance fv somewhat homomorphic encryption on gpus: An implementation using cuda," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 70–95, 2018.

[32] 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 Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1330–1343, 2020.

[33] J. Jang, Y. Lee, A. Kim, B. Na, D. Yhee, B. Lee, J. H. Cheon, and S. Yoon, "Privacy-preserving deep sequential model with matrix homomorphic encryption," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 377–391.

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

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

[36] O. Ore, *Number theory and its history*. Courier Corporation, 1988.

[37] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.

[38] Q. Lou and L. Jiang, "She: A fast and accurate deep neural network for encrypted data," *Advances in neural information processing systems*, vol. 32, 2019.