IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 September 2023, accepted 19 September 2023, date of publication 22 September 2023,
date of current version 28 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3318433

RESEARCH ARTICLE

Optimizations of Privacy-Preserving DNN for Low-Latency Inference on Encrypted Data

HYUNHOON LEE, (Graduate Student Member, IEEE), AND YOUNGJOO LEE, (Senior Member, IEEE)
Department of Electrical Engineering, Pohang University of Science and Technology, Pohang 37673, South Korea

Corresponding author: Youngjoo Lee (youngjoo.lee@postech.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean Government Ministry of Science and ICT (MSIT), South Korea (Development of high-speed encryption data processing technology that guarantees privacy based hardware) under Grant 2021-0-00779; and in part by MSIT, under the Information Technology Research Center (ITRC) Support Program supervised by IITP under Grant IITP-2021-2020-0-01461 and Grant IITP-2023-2020-0-01612.

ABSTRACT Homomorphic encryption (HE) based on the CKKS scheme is a promising candidate for implementing privacy-preserving deep neural networks (PP-DNN) by performing operations directly on the encrypted data. However, due to the computational complexity of HE operation, even simple PP-DNNs require a huge amount of processing time. In order to reduce the processing time of PP-DNN, in this paper, we present an innovative, low-latency model optimization solution for PP-DNNs. Our proposed low-latency model optimization solution exploits second-order polynomials that approximate original activation functions, ensuring low-latency and accurate DNN performance. To further reduce the processing latency of PP-DNNs, we introduce the coefficient absorbing technique and a masking convolution for convolutional layers. The experimental results show that the proposed solution constructs bootstrapping-free PP-DNN and reduces the inference latency of CKKS-based ResNet-34 by 35% in the CIFAR-100 dataset and ResNet-32 by 77% in the CIFAR-10 dataset compared to previous approaches while maintaining the same level of inference accuracy. Moreover, through the layer-wise latency analysis, we show the efficacy of our approaches, and through validation in various scenarios, we demonstrate the generality of our methods.

INDEX TERMS RNS-CKKS, convolutional neural network, homomorphic encryption, privacy preserving neural network.

I. INTRODUCTION

Due to the superior algorithmic performance, deep neural networks (DNNs) have been actively applied to the practical applications associated with numerous multimedia data such as images [1], [2], [3], [4], videos [5], [6], [7], and speeches [8], [9]. As the computational complexity of recent DNN models has continuously increased for better accuracy, the contemporary DNN-based services have been generally realized at the server-scale computing platforms; and therefore, the client should transfer its data to the dedicated server to get the desired DNN results [10].

For privacy-critical applications, however, it is impossible to directly open sensitive user-level information such as personal images, secret messages, or biomedical data. In order to apply advanced DNN solutions even to these sensitive data, the concept of privacy-preserving DNNs (PP-DNNs) based inference has recently gained great notability and popularity. To implement the PP-DNN, there are several key concepts exist, such as differential privacy (DP), secure multi-party computation (MPC), and homomorphic encryption (HE). DP ensures the privacy of data by adding noise to individual data, preserving sensitive information while allowing accurate model training and analysis. By incorporating DP, it ensures that the outputs of models do not reveal sensitive information about the data used in the training process [11]. However, the effectiveness of the DP-based

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

PP-DNN method on inference is still not revealed. The MPC enables multiple parties to collaboratively train or infer a machine-learning model on their respective datasets without the need to share the raw data. MPC ensures that the data of each party remains encrypted throughout the computation [12]. But, the MPC-based approach increases the communication overheads between parties, which cannot be acceptable, especially for energy-limited edge-level clients. HE enables computations directly on encrypted data, allowing machine learning models to be trained and inferred on encrypted data without decryption. This approach ensures data privacy is maintained throughout the entire computation, even during model training and evaluation [13]. However, the HE-based method is computationally intensive, thereby causing a substantial increase in computational overhead.

To implement PP-DNN-based inference, we focus on the HE-based approach. As mentioned earlier, it is essential to note that HE provides strong privacy guarantees but requires a considerable computational burden in the inference process. Due to such a burden, at least 200ms of latency is required for multiplication between ciphertexts [14], and, furthermore, the state-of-the-art PP-DNN implementation based on HE still takes more than 38 minutes to process a ResNet-20 model, even in a server-scale environment [15]. The latency caused by such extreme computational demands makes it infeasible for practical application. In this paper, we propose an optimization technique to reduce the performance gap drastically.

In the emerging CKKS scheme [16], the complexity of each operation is related to the ciphertext level l, and l also determines the consecutive multiplications, also known as multiplicative depth. As computational complexity is related to l, simply increasing l to achieve a higher multiplicative depth results in excessive delay. Instead, it is possible to allow unlimited multiplicative depth by activating the bootstrapping steps [17], which refresh the current level l, i.e., multiplicative depth. However, it is hard to freely increase the number of serialized multiplications due to the bootstrapping itself incurring long processing delays. Therefore, to achieve low-latency inference, it is important to address the issue of latency related to multiplicative depth. Consequently, it is essential to develop optimization techniques to reduce the required multiplicative depth. By doing so, the predefined maximum level, denoted as l_{MAX}, and the number of bootstrapping are effectively decreased, thereby leading to a reduction in overall computational complexity and resulting in decreased latency. For practical PP-DNN processing, therefore, previous research has focused on developing HE-aware CNN architectures that minimize the multiplicative depth while supporting sufficiently acceptable recognition accuracy [15], [18], [19], [20], [21], [22], [23], [24], [25], [26]. However, to avoid or minimize using bootstrapping, they implement relatively shallow networks [18], [19], [20], [21], [22], [23], [25], [26]. Alternatively, there exist studies on deep networks [15], [24]; however, they require an excessive number of bootstrapping.

In this paper, we newly develop advanced HE-aware model optimization techniques, significantly relaxing the required multiplicative depth of PP-DNN processing and achieving low latency PP-DNN-based inference. Contributions of this paper are summarized as follows:

- The proposed work adopts approximated second-order polynomials based on the original non-linear activation functions to achieve low latency without degrading the recognition accuracy of original models.
- The coefficient absorbing and masking convolution schemes are proposed to construct the HE-aware convolution layers, further reducing the processing time.
- Experimental results reveal that the proposed approaches even lead to bootstrapping-free ResNet-32 and ResNet-34 with feasible HE parameters while maintaining the baseline accuracy, reducing the processing delay by 4.3 and 1.5 times in the CIFAR-10 and CIFAR-100 datasets, respectively, compared with the state-of-the-art PP-DNN studies.
- We evaluate the proposed work using various datasets, such as Fashion-MNIST, CIFAR-10, and CIFAR-100, various networks, and various HE libraries, and the results show that the proposed work can be applied generally.

The rest of this paper is organized as follows. Section II describes the related works and background for HE and PP-DNN, and the proposed optimization methods for low-latency PP-DNN operations are described in Section III. Experimental results on the practical CNN models are evaluated and compared with the other works in Section IV, and the conclusion remarks are finally made in Section V.

II. BACKGROUNDS

A. RELATED WORKS

Although HE provides the highest level of security, the HE can only perform addition and multiplication operations. So, using HE to compute the non-linear layers in a conventional DNN is challenging. Conceptually, depending on how evaluating the non-linear layers, HE-based PP-DNN architectures are categorized into interactive or non-interactive schemes [15], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], as shown in Fig. 1. The interactive approach resolves the challenge of computing non-linear activations through MPC, where the client and server interact to compute non-linear functions on the data [27], [28], [29], [30], [31], [32]. More specifically, the server returns the intermediate results of each convolution layer, and the client performs the non-linear operations on the received data based on the MPC protocol. Then, the results of non-linear operations are re-transmitted to the server to continue the following linear operations as depicted in Fig. 1 (a). The interactive method may totally remove the time-consuming bootstrapping steps by returning

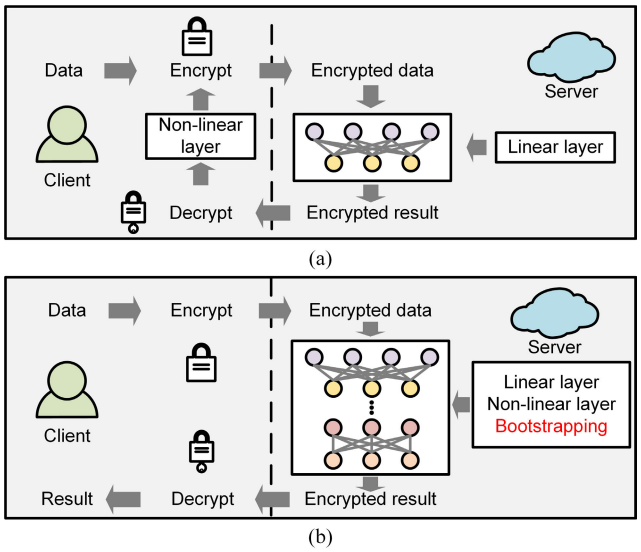

FIGURE 1. Conceptual diagram of (a) interactive PP-DNN and (b) non-interactive PP-DNN.

the intermediate ciphertexts. However, they should increase the client-server communication overheads that cannot be acceptable, especially for energy-limited edge-level clients.

On the other hand, the recent non-interactive approach introduces the end-to-end PP-DNN processing at the server so that the client only receives the final ciphertext results [15], [18], [19], [20], [21], [22], [23], [24], [25], [26]. To evaluate the non-linear layer in the HE domain, several works replace the non-linear layer with low-degree polynomials, such as square [18], [19] and second-order polynomials [20], [21], [22], [23], [25], [26], that can be computed with HE operations. However, they remain in implementing relatively shallow networks due to the degraded recognition accuracy in deep networks when using second-order polynomials. More specifically, the first HE-based PP-DNN [18] implemented only 2-linear layer networks with square activation. Subsequent works are also performed at a shallow layer [19], [20], [21], [23], [25]. Reference [22] implemented InceptionNet with second-order polynomial activation and proposed finding a low-latency architecture algorithm, but they use a relatively light network, which can not achieve sufficient accuracy. Reference [26] implemented ResNet-20 for CIFAR-10 with second-order polynomial activation, but they require time-consuming bootstrapping. Alternatively, the high-order polynomial approximation has been applied to convert the previous non-linear activation functions not to severely degrade the recognition accuracy [15], [24], [33]. Due to the excessively-increased multiplicative depth from the high-order polynomials, targeting the practical CNN models, we have to actively insert bootstrapping steps to refresh the multiplicative depth as depicted in Fig. 1 (b). In more detail, [15], [24] used precisely approximated ReLU function, a 29-degree polynomial, as non-linear activation and successfully implement ResNet-20 and ResNet-110, respectively. However, bootstrapping is required for each non-linear layer, and bootstrapping consumes more than half of the total execution time, resulting in excessively long runtimes.

Moreover, several works propose optimization techniques for the non-interactive PP-DNN. References [15] and [23] proposed a multiplexed convolution method, which utilizes the non-valid slots generated by pooling or convolution with a stride of 2 or more to reduce the multiplication in the convolutional layer. References [15], [22], and [25] proposed a merging method, which merges adjacent linear layers of neural networks, such as the convolutional layer and batch normalization layer. This merging method can decrease the required multiplicative depth without affecting the results. As a result, l_{MAX} is reduced; thus, the overall latency, which is related to l, is reduced. The merging method is a simple yet powerful technique for achieving low-latency PP-DNN. Despite their optimization technique, their work still remains in shallow networks, which cannot achieve acceptable accuracy using in practical cases, or requires bootstrapping, which is a time-consuming operation. In this work, focusing on the non-interactive PP-DNN, we propose a more drastic level reduction method, which can even implement bootstrapping-free ResNet-34 inference.

B. HOMOMORPHIC ENCRYPTION

As illustrated in Fig. 1, applying HE schemes allows to directly perform arithmetic operations without requiring pre-decryption steps in the traditional encryption systems [16], [34]. The ciphertext in HE is defined as two N degree polynomials in an integer ring. The encryption process encrypts the $N/2$ size vector into the ciphertext with $N/2$ slots. For ease of explanation, we represent the encryption of $N/2$ size vector $\mathbf{v}$ as $[[\mathbf{v}]]$. Then, the well-known HE operations have been developed from numerous studies as follows [16].

- Addition: $[[\mathbf{v}_0]] \oplus [[\mathbf{v}_1]] = [[\mathbf{v}_0 + \mathbf{v}_1]]$
- Plaintext multiplication: $[[\mathbf{v}_0]] \odot \mathbf{v}_1 = [[\mathbf{v}_0 \cdot \mathbf{v}_1]]$
- Ciphertext multiplication: $[[\mathbf{v}_0]] \otimes [[\mathbf{v}_1]] = [[\mathbf{v}_0 \cdot \mathbf{v}_1]]$
- Rotation: $rot([[\mathbf{v}]], r) = [[\mathbf{v} \ll r]]$

Where $\oplus$, $\odot$, and $\otimes$ represent addition, plaintext multiplication, and ciphertext multiplication in the ciphertext domain, respectively, and $+$ and $\cdot$ represent element-wise addition and multiplication in the vector domain, respectively. The rotation rot uses the rotation amount of r, rearranging coefficients of ciphertext $[[\mathbf{v}]]$. This operation, i.e., $\mathbf{v} \ll r$ operation in vector domain, rotates the data in slots $(v[0], v[1], \ldots, v[N/2-1])$ to $(v[r], v[r+1], \ldots, v[N/2-1], v[0], \ldots, v[r-1])$.

C. PP-DNN BASED ON RNS-CKKS

1) RNS-CKKS

Among various HE schemes, in this work, we focus on the recent RNS-CKKS-base HE [35], which is regarded as a practical solution for constructing PP-DNN models [15],

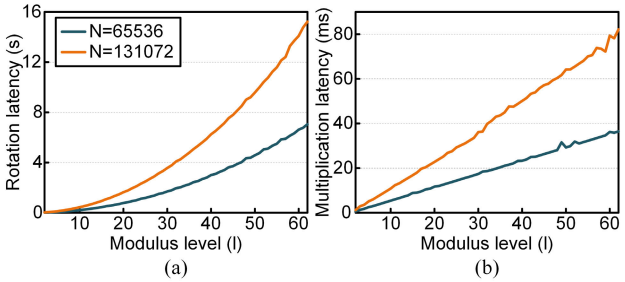

FIGURE 2. Latency of (a) rotation and (b) plaintext multiplication according to N and l.

[20], [21], [22] due to its property that enables fixed-point arithmetic. CKKS scheme enables the fixed-point arithmetic by encoding the $N/2$ size fixed-point vector into an N-degree polynomial, and the ciphertext is generated by performing the encryption process on this polynomial. We represent this encoding of vector $\mathbf{v}$, i.e., the plaintext, as $[\mathbf{v}]$. Further, the RNS-CKKS scheme reduces the computational complexity of CKKS by using the property of the Chinese Remainder Theorem (CRT), which can replace the long coefficient size with practical machine word size coefficients. The l-level polynomial of RNS-CKKS is the element of the polynomial ring $\mathbb{Z}_{Q_l}[x]/\langle X^N+1\rangle$, where $Q_l = \Pi_{i=0}^{l} q_i$ and each q_i is a prime number.

During the encoding process, the scaling factor is introduced to represent the fixed-point vector in the plaintext and ciphertext, and the HE multiplication increases this scaling factor. To precisely control the scaling factor, the rescaling operation exists. When considering plaintext multiplication between $[[\mathbf{v}_0]]$ and $[\mathbf{v}_1]$, each scaling factor being Δ_0 and Δ_1, the resulting ciphertext $[[\mathbf{v}_0 \cdot \mathbf{v}_1]]$ has a scaling factor of $\Delta_0 \times \Delta_1$. To decrease the scaling factor, the rescaling operation is performed by dividing the prime q_l. As a result, the scaling factor is reduced to $\frac{\Delta_0 \times \Delta_1}{q_l}$ and the level of ciphertext is reduced from l to $l-1$. Because this rescaling operation can only be performed l_{MAX} times, the multiplicative depth is defined as l_{MAX}. Bootstrapping [17] can refresh the reduced level to the maximum level l_{MAX}, but it requires a significant execution time.

2) MULTIPLICATIVE DEPTH OF PP-DNN

Level l determines the computational complexity of HE operations. More specifically, the complexity of plaintext multiplication is $\mathcal{O}(N \cdot l)$ and complexity of rotation is $\mathcal{O}(N \cdot \log N \cdot l^2)$, as shown in Fig. 2. As stated in [22], the multiplicative depth of PP-DNN is defined by the number of serialized HE multiplications in the network, and the maximum level l_{MAX} is selected based on the multiplicative depth of PP-DNN. Therefore, the complexity of PP-DNN can be defined as $\mathcal{O}(N \cdot \log N \cdot l_{\mathrm{MAX}}^2)$, similar to the most complex operation, which is rotation. Reducing N and l_{MAX} can decrease the complexity of PP-DNN. However, as the security level is determined by N and l_{MAX}, to keep the security level, N should be large enough depending on

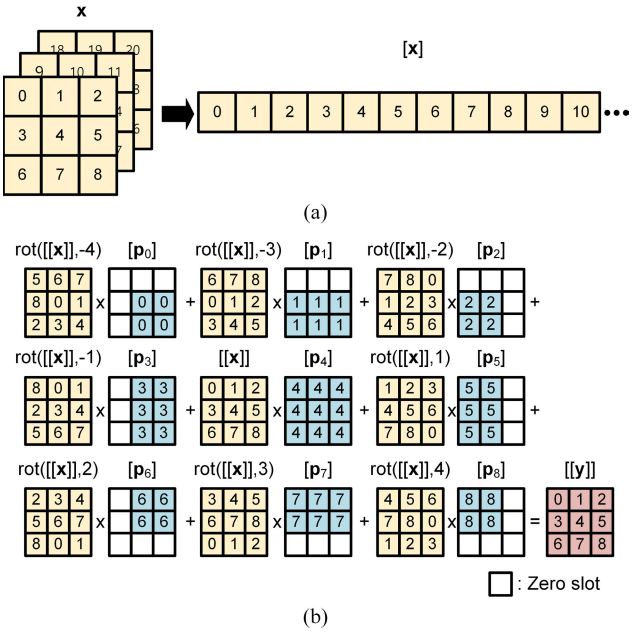

FIGURE 3. (a) Mapping from tensor to plaintext and (b) efficient homomorphic convolution method from [27].

l_{MAX}. So, to maintain the bit security level while reducing the complexity of PP-DNN, it is necessary to decrease the multiplicative depth of PP-DNN, as mentioned in [22].

D. CONVOLUTION ON HOMOMORPHIC ENCRYPTION

The operation of the convolutional layer on CNN is defined as:

$$\mathbf{y}[o][h][w] = \sum_{c,j,n} (\mathbf{x}[c][h+j][w+n] \cdot \mathbf{w}[o][c][j][n]) + \mathbf{q}[o], \tag{1}$$

where $\mathbf{x}$, $\mathbf{w}$, $\mathbf{q}$, and $\mathbf{y}$ are input, weight, bias, and output tensors, and o, c, h, w, j, and n are the index of the output channel, input channel, image height, image width, kernel height, and kernel width, respectively. This convolution operation can be performed parallel by the SIMD property of HE. Reference [27] proposed efficient homomorphic convolution that performs SIMD convolution operation in the ciphertext domain for one or multiple channels according to the packing method. As shown in Fig. 3 (a), [27] treated 2 (or 3)-dimension tensor as a 1-dimension vector and encrypt this vector to ciphertext. $\mathbf{x}$ is the input tensor, and each number represents the element index of $\mathbf{x}$, i.e., $\mathbf{x}[0], [0], [0]$, $\mathbf{x}[0], [0], [1], \ldots$ are represented as $0, 1, \ldots$ in the figure. Fig. 3 (b) shows the 3×3 convolution method on HE proposed in [27] for the 3×3 input with one input and one output channel. $[[\mathbf{x}]]$ and $[[\mathbf{y}]]$ represent the ciphertexts of the input tensor $\mathbf{x}$ and output tensor $\mathbf{y}$, respectively. $[\mathbf{p}_0], [\mathbf{p}_1], \ldots, [\mathbf{p}_8]$ are weight plaintexts, which consist of weight values of the 3×3 kernel $\mathbf{w}[0], [0], [0][0]$, $\mathbf{w}[0], [0], [0][1], \ldots, \mathbf{w}[0], [0], [2], [2]$, respectively. Suppose each channel of the input tensor is encrypted separately into different ciphertexts. In that case, each HE multiplication

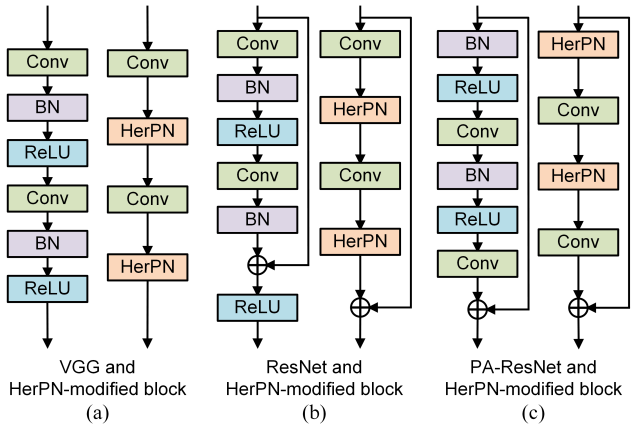

FIGURE 4. The modified models with HerPN block of (a) VGG net, (b) ResNet, and (c) PA-ResNet.

can perform $W \cdot H$ multiplications in parallel, where W and H represent the image width and height, respectively. Alternatively, if multiple channels are encrypted into a single ciphertext, each HE multiplication can perform $W \cdot H \cdot C$ multiplications in parallel, where C is the number of channels in the ciphertext. To efficiently pack the multiple channels into the ciphertext, this method fills appropriate slots in the weight plaintexts with zeros to allow padded convolution without additional overhead. It enables more channels to be accommodated in a single ciphertext, increasing the capacity from $\lfloor \frac{N}{2\cdot(W+2\cdot P)\cdot(H+2\cdot P)} \rfloor$ to $\lfloor \frac{N}{2\cdot W\cdot H} \rfloor$, where P is the padding size. The HE rotation is performed to align the data slots during the convolution operation.

E. HerPN BLOCK

The effect of replacing the activation function with a Hermite polynomial is analyzed in [36] and [37]. Basically, using Hermite polynomial as an activation function instead of ReLU enables more stable training and achieves similar accuracy. Further, [32] proposed the HerPN block, which utilizes the Hermite expansion of ReLU as a activation function with basis-wise normalization, to use in PP-DNN. This method preserves the accuracy of the backbone networks which utilize non-linear activations such as ReLU. In this section, we provide a brief overview of the HerPN block introduced in [32].

Hermite expansion of ReLU is defined as follows:

$$\mathrm{ReLU}(x) = \sum_{i=0}^{\infty} \hat{f}_i h_i(x) \tag{2}$$

where $h_n(x) = \frac{1}{\sqrt{n!}}(-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n} e^{\frac{-x^2}{2}}$ and $\hat{f}_i$ is:

$$\hat{f}_i = \begin{cases} \frac{1}{\sqrt{2\pi}} & n = 0, \\ \frac{1}{2} & n = 1, \\ 0 & n \geq 2 \text{ and odd}, \\ \frac{((n-3))!!)^2}{\sqrt{2\pi n!}} & n \geq 2 \text{ and even} \end{cases} \tag{3}$$

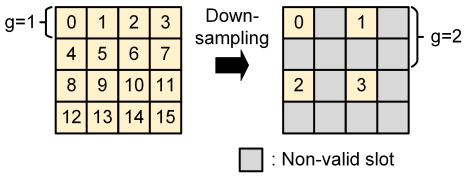

FIGURE 5. Example of non-valid slots generated from down-sampling operation with $s = 2$.

Then, to apply basis-wise normalization, batch normalization is performed to the output of the Hermite polynomial. Finally, the HerPN block for input x is as follows:

$$f(x) = \sum_{i=0}^{d} \hat{f}_i \frac{h_i(x) - \mu}{\sqrt{\sigma^2 + \epsilon}}. \tag{4}$$

The HerPN block generated from the above process can replace both the batch normalization and ReLU layers in a neural network while preserving the accuracy of the original models. The Modified models with HerPN block for VGG [1], ResNet [2], and PA-ResNet [3] are shown in Fig. 4 (a), (b), and (c), respectively.

F. THREAT MODEL

In our paper, the threat model is similar to the previous works on PP-DNNs [15], [22]. The client encrypts the private data using HE and sends it to the semi-honest server, i.e., the server performs computation accurately but is curious about the client data. The server performs private inference on the encrypted data without decrypting it by the property of HE. Then, the server returns the ciphertext of the inference result to the client. Only the client that holds the secret key can decrypt the result using the secret key.

III. PROPOSED METHOD

In this section, we introduce the techniques of preserving accuracy while reducing multiplicative depth through a second-order polynomial, coefficient absorbing, and masking convolution. First, to implement non-linear layers in PP-DNN, we replace the ReLU with a second-order polynomial using the HerPN block. By employing a second-order polynomial as the activation function, it is possible to drastically reduce the multiplicative depth compared to using higher-order approximate polynomials, thereby enabling the elimination of bootstrapping. However, conventional approaches using quadratic polynomials encounter an issue of accuracy degradation as the network deepens. To address this, we adopt the HerPN block explained in Section II-E as our activation. Even by utilizing only the 0th to 2nd term of the HerPN block, the accuracy of networks using ReLU as the activation function can be preserved [32].

Then, before explaining the proposed absorbing techniques, we first describe the gap g between the valid slots and the conventional merging method. In the non-interactive PP-DNN scenario, non-valid slots occur due to down-sampling operations such as pooling or convolution operations with

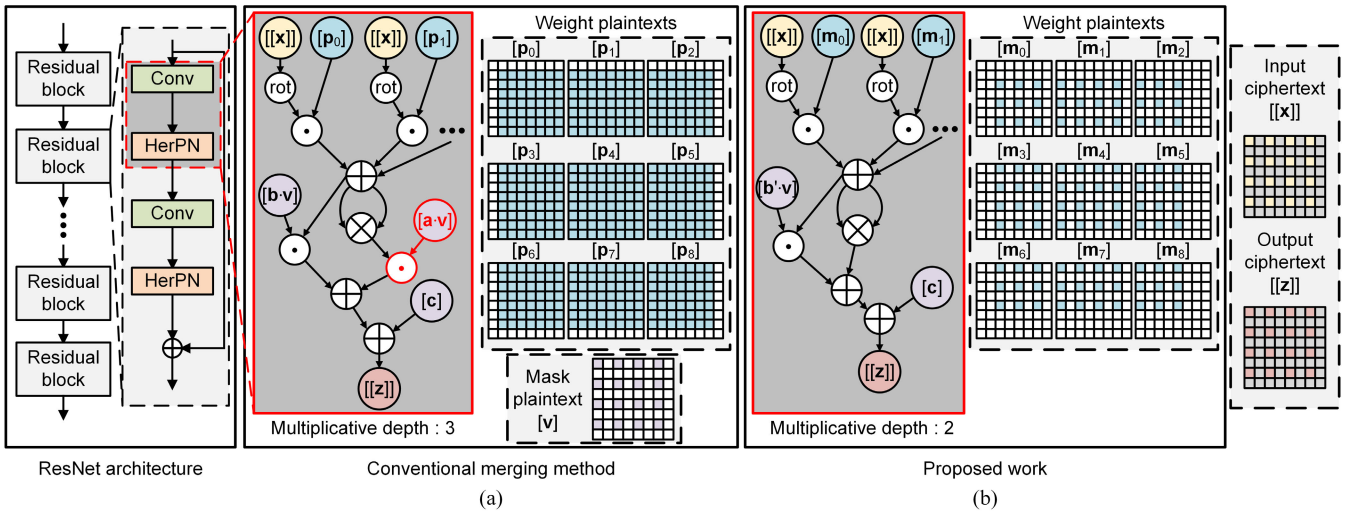

FIGURE 6. (a) Conventional merging method and (b) proposed work to reduce multiplicative depth for ResNet architecture when $g = 2$ and $s = 1$.

$s \geq 2$, where s is the stride of operations, as shown in Fig. 5. The number of non-valid slots is $g^2 - 1$ times larger than valid slots. g expands by s times through down-sampling operations.

Fig. 6 shows the difference between the conventional merging method and the proposed work when using second-order polynomial as non-linear activation. We assume that the initial input image size is 8×8 and the input ciphertext $[[\mathbf{x}]]$ of the current layer is 4×4 size and $g = 2$ due to the previous down-sampling. When stride-1 3×3 convolution and HerPN block operation are performed on $[[\mathbf{x}]]$, the output ciphertext $[[\mathbf{z}]]$ has the same size 4×4 and $g = 2$ as the input. The conventional merging method merges the batch normalization layer into the non-linear layer or convolutional layer, achieving a multiplicative depth of 3 for the convolutional layer followed by the non-linear layer [15], [22], [23], [25]. To handle non-valid slots, previous works have used additional masking layers or merged the masking patterns into the coefficient of the second-order term of activation layers, as shown in Fig. 6 (a), where $\mathbf{a}$ is filled with the coefficient of the second-order term and $\mathbf{v}$ consists of 0 and 1 values to mask non-valid slots. As $s = 1$ and $g = 2$, $\mathbf{v}$ has the same valid slots as the input ciphertext $[[\mathbf{x}]]$. By multiplying $[\mathbf{a} \cdot \mathbf{v}]$ with the output of the convolution layer, the non-valid slots are masked to zero.

To eliminate this masking layer, we propose absorbing techniques of the vectors $\mathbf{a}$ and $\mathbf{v}$ into the prior convolutional layer. To absorb the vector $\mathbf{a}$, coefficient absorbing is proposed in Section III-A. Then, to eliminate the mask vector $\mathbf{v}$ multiplication, masking convolution is proposed in Section III-B. The proposed masking convolution generalizes the homomorphic convolution method of [27] for $g \geq 2$ and generates masked weight plaintexts accordingly. So, the weight plaintexts for 3×3 convolution is changed from $[\mathbf{p}_i]$ in Fig. 6 (a) to $[\mathbf{m}_i]$ in Fig. 6 (b). By combining the proposed methods, the multiplicative depth of each convolution and HerPN block pair can be further reduced compared to the previous approach, as shown in Fig. 6 (b), and thus the total multiplication depth of the network is also reduced. Therefore, as the total multiplicative depth of the network is decreased significantly, the maximum level l_{MAX} is also decreased. So, the latency of PP-DNN, the complexity of which can be represented as $\mathcal{O}(N \cdot \log N \cdot l_{\mathrm{MAX}}^2)$, decreases as the level l_{MAX} decreases. Finally, Section III-C describes the method for applying the proposed methods to the channel packing.

A. COEFFICIENT ABSORBING

The purpose of coefficient absorbing is to absorb the coefficients of the second-order term in the polynomial activation function into the previous convolution layer, thereby replacing the operation of the activation layer from the conventional $ay^2 + by + c$ to $\hat{y}^2 + b'\hat{y} + c$. Therefore, the multiplicative depth of the activation layer is reduced.

The proof of coefficient absorbing, when utilizing the HerPN block with three bases, h_0, h_1, and h_2, as activation function, proceeds according to the subsequent explanation. HerPN block with three bases, i.e., $d = 2$ in Eq.(5), is as follows:

$$\begin{aligned} z &= \sum_{i=0}^{2} \hat{f}_i \frac{h_i(y) - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \\ &= \hat{f}_0 \frac{1 - \mu_0}{\sqrt{\sigma_0^2 + \epsilon}} + \hat{f}_1 \frac{y - \mu_1}{\sqrt{\sigma_1^2 + \epsilon}} + \hat{f}_2 \frac{\frac{y^2 - 1}{\sqrt{2}} - \mu_2}{\sqrt{\sigma_2^2 + \epsilon}}, \end{aligned} \tag{5}$$

where $\hat{f}_i$ are from Eq. (3), and μ_i and σ_i are the mean and standard deviation of each basis, respectively, y is the output of the convolution layer, and z is the output of HerPN block. Eq. (5) is equivalent to $ay^2 + by + c$ when we set $\hat{\sigma}_i = \sqrt{\sigma_i^2 + \epsilon}$, $a = \frac{\hat{f}_2}{\sqrt{2}\hat{\sigma}_2}$, $b = \frac{\hat{f}_1}{\hat{\sigma}_1}$,

and $c = \hat{f}_0 \frac{1-\mu_0}{\hat{\sigma}_0} - \hat{f}_1 \frac{\mu_1}{\hat{\sigma}_1} - \hat{f}_2 \frac{1+\sqrt{2}\mu_2}{\sqrt{2}\hat{\sigma}_2}$. Finally, we absorb the coefficient of highest order term as follows:

$$\begin{aligned} z &= ay^2 + by + c \\ &= (\sqrt{a}y)^2 + \frac{b}{\sqrt{a}}(\sqrt{a}y) + c \\ &= \hat{y}^2 + b'\hat{y} + c, \end{aligned} \tag{6}$$

where $\hat{y} = \sqrt{a}y$ and $b' = \frac{b}{\sqrt{a}}$. The coefficient of the second-order term is absorbed in this way, and the weights and biases of the convolutional layer are multiplied by the absorbed $\sqrt{a}$. Consequently, the convolution operation represented in Eq. (1) is modified as follows:

$$\begin{aligned} \hat{\mathbf{y}}[o][h][w] &= \sqrt{a} \cdot \mathbf{y}[o][h][w] \\ &= \sum_{c,j,n} (\mathbf{x}[c][h+j][w+n] \cdot (\sqrt{a} \cdot \mathbf{w}[o][c][j][n])) \\ &\quad + (\sqrt{a} \cdot \mathbf{q}[o]) \\ &= \sum_{c,j,n} (\mathbf{x}[c][h+j][w+n] \cdot \hat{\mathbf{w}}[o][c][j][n]) + \hat{\mathbf{q}}[o], \end{aligned} \tag{7}$$

where $\hat{\mathbf{w}}[o][c][j][n] = \sqrt{a} \cdot \mathbf{w}[o][c][j][n]$ and $\hat{\mathbf{q}}[o] = \sqrt{a} \cdot \mathbf{q}[o]$ and all other parameters follow the representation of Eq. (1). Through this proof, the operation of the activation layer can be successfully replaced from $ay^2 + by + c$ to $\hat{y}^2 + b'\hat{y} + c$, and therefore, we theoretically demonstrated that coefficient absorbing allows for reducing the multiplication depth while obtaining equivalent results to the conventional approach.

In Fig. 6, [**a**], [**b**], [**c**], and [**b**′] consist of a, b, c, and b' in Eq. (6), respectively. [$\mathbf{p}_i$] are weight plaintexts consisting of weight tensor $\mathbf{w}$ before absorption, i.e., $\mathbf{w}[o][c][j][n]$ in Eq. (1), following the method of [27]. The weight plaintexts [$\mathbf{m}_i$] consist of absorbed weight tensor $\hat{\mathbf{w}}$, i.e., $\hat{\mathbf{w}}[o][c][j][n]$ in Eq. (7), and zero values, through masking convolution, described in the next section. Note that our method absorbs the parameters of neural networks to the prior layer, so the result of neural networks remains in the same values and does not degrade accuracy.

B. MASKING CONVOLUTION

By further developing the convolution method of [27], which generates weight plaintexts for homomorphic convolution, we propose a generalized masked weight vector generation method for any stride s and gap g, as described in Algorithm 1. In this algorithm, n_{ch} and n_{cw} are the height and width of the current feature map, respectively, which are reduced by the down-sampling operation and calculated as denoted in the initialization step. n_{ih} and n_{iw} are the initial size of the input image height and width, respectively. k_h and k_w are kernel size, s is stride, and $\hat{\mathbf{w}}$ is the $k_h \times k_w$ absorbed weight tensor of the current convolutional layer. The masked weight vectors $\mathbf{m}_i$ generated by this algorithm are composed of a masking pattern consisting of zeros and absorbed weight values, enabling the omission of the process of multiplying by the mask plaintext [**v**]. The generated masked weight vectors $\mathbf{m}_i$ are then encoded into plaintexts [$\mathbf{m}_i$]. Each [$\mathbf{m}_i$] only has valid values in first $n_{ih} \times n_{iw}$ slots, and other slots, i.e., $N/2 - n_{ih} \times n_{iw}$ slots, remain in zero. Using the [$\mathbf{m}_i$], the homomorphic convolution on [[**x**]] performs as follows:

$$[[\hat{\mathbf{y}}]] = \sum_{i=-t_h}^{t_h} \sum_{j=-t_w}^{t_w} rot([[\mathbf{x}]], -g(n_{iw}i + j)) \odot [\mathbf{m}_{k_w(t_h+i)+t_w+j}], \tag{8}$$

where $t_w = \lfloor k_w/2 \rfloor$ and $t_h = \lfloor k_h/2 \rfloor$. We assume that the convolution operation performs on padded features and omits the representation for channels. We represent Eq. (8) using $\circledast$ as follows:

$$[[\hat{\mathbf{y}}]] = [[\mathbf{x}]] \circledast [\mathbf{M}], \tag{9}$$

where $[\mathbf{M}] = ([\mathbf{m}_0], [\mathbf{m}_1], \ldots, [\mathbf{m}_{k_h k_w - 1}])$.

Algorithm 1 Masked Weight Vector Generation

Input:
Current height n_{ch} and width n_{cw},
input height n_{ih} and width n_{iw},
kernel size k_h and k_w, gap g, stride s,
and absorbed weight tensor $\hat{\mathbf{w}} \in \mathbb{R}^{k \times k}$

Output:
Masked weight vectors
$\mathbf{M} = (\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_{k_h k_w - 1})$, where $\mathbf{m}_i \in \mathbb{R}^{N/2}$

Initialization:
Padding size $P_h = \lfloor k_h/2 \rfloor$
Padding size $P_w = \lfloor k_w/2 \rfloor$
$\mathbf{m}_i = \mathbf{0}$, for $i \in [0, k_h k_w - 1]$
$n_{ch} = \lfloor \frac{n_{ch} - k_h + 2 \cdot P_h}{s+1} \rfloor$
$n_{cw} = \lfloor \frac{n_{cw} - k_w + 2 \cdot P_w}{s+1} \rfloor$

Processing:
```
for i = 0 to k_h − 1 do
    f_h = ⌊k_h/2⌋ − i
    for j = 0 to k_w − 1 do
        f_w = ⌊k_w/2⌋ − j
        for l = 0 to l < n_ch − 1 do
            h_pos = l − n_ih/g
            if 0 ≤ f_h ≤ l or h_pos < f_h < 0 then
                for n = 0 to n < n_cw − 1 do
                    w_pos = n − n_iw/g
                    if 0 ≤ f_w ≤ n or w_pos < f_w < 0
                     then
                        o = (l · n_iw + n) · g · s
                        m_{k_w·i+j}[o] = ŵ[i][j]
                    end
                end
            end
        end
    end
end
M = (m_0, m_1, ..., m_{k_h k_w − 1})
return M
```

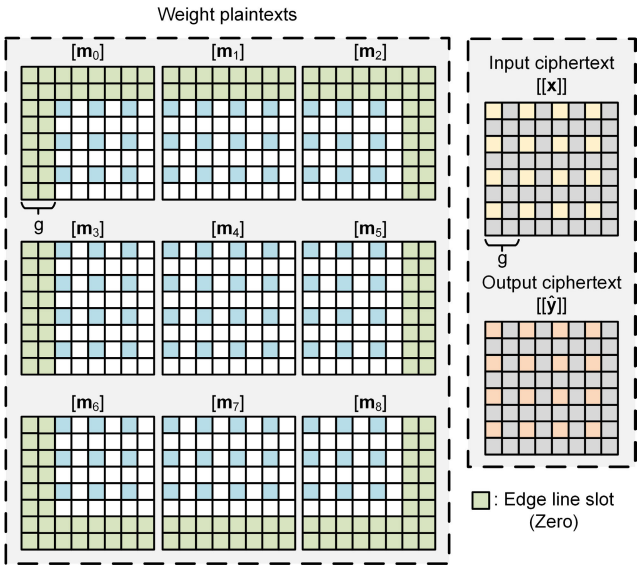

FIGURE 7. The slots of weight plaintexts, input ciphertext, and output ciphertext when $g = 2$ and $s = 1$.

For instance, when $k_h = 3$, $k_w = 3$, $n_{ih} = 8$, $n_{iw} = 8$, n_{ch}, $n_{cw} = 4$, $g = 2$, $s = 1$, and both input and output channels 1, weight plaintexts are generated, as shown in Fig. 7, according to Algorithm 1. To implement stride one convolution for input ciphertext $[[\mathbf{x}]]$ with $g = 2$, the zero and the weight values are alternately placed in the slots of weight plaintexts. Additionally, to solve the slot mismatch caused by non-valid slots, there are exist zero value slots at the edges, as shown in green-colored slots in the figure, and $[[\mathbf{x}]]$ needs to be rotated by two times as many slots compared to the conventional method. So, the homomorphic convolution of this example is defined as follows:

$$[[\hat{\mathbf{y}}]] = \sum_{i=-1}^{1}\sum_{j=-1}^{1} rot([[\mathbf{x}]], -2(8i+j)) \odot [\mathbf{m}_{3(1+i)+1+j}]. \tag{10}$$

The output ciphertext $[[\hat{\mathbf{y}}]]$ has the same non-valid slots as the input ciphertext $[[\mathbf{x}]]$ because $s = 1$. Using the generated $[\mathbf{m}_i]$, convolutional and activation layers can be performed with the multiplicative depth of 2 without masking layer, as shown in Fig. 6 (b). Algorithm 1 assumes both the input and output channels are 1, but it can be easily generalized to various hyperparameters of PP-DNN.

C. CHANNEL PACKING

Similar to previous works [15], [20], [21], [22], [23], [27], we adopt the channel packing method, which packs $c_n = \lfloor \frac{N}{2 \cdot W \cdot H} \rfloor$ channels per ciphertext, to achieve low latency inference. Each layer of the network has $\lceil \frac{c_i}{c_n} \rceil$ input ciphertexts and $\lceil \frac{c_o}{c_n} \rceil$ output ciphertexts, where c_i and c_o are the numbers of input and output channels, respectively. Fig. 8 shows the homomorphic convolution method using channel packing, where $\circledast$ is the operation defined in Eq. (9), $[[\mathbf{x}_{\text{pack}}]]$ represents the packed ciphertext, i.e., $\mathbf{x}_{\text{pack}}$ is

TABLE 1. The number of rotations and multiplications according to packing method.

	Rotation	Multiplication
w/o packing	$(k_h \cdot k_w - 1) \cdot c_i$	$k_h \cdot k_w \cdot c_i \cdot c_o$
w/ packing	$(k_h \cdot k_w - 1) \cdot \frac{c_i}{c_n} + (c_n - 1) \cdot \frac{c_o}{c_n}$	$k_h \cdot k_w \cdot \frac{c_i}{c_n} \cdot c_o$
Multiplexed packing [15]	$(k_h \cdot k_w - 1) \cdot \frac{c_i}{c_n \cdot g^2} + (c_n - 1) \cdot \frac{c_o}{c_n} + \lceil \frac{c_i}{c_n \cdot g^2} \rceil \cdot (g^2 - 1) + \lceil \frac{c_o}{c_n} \rceil \cdot \lceil \log(g^2) \rceil$	$k_h \cdot k_w \cdot \frac{c_i}{c_n \cdot g^2} \cdot c_o$

consist of concatenated c_n channels of input, each $\mathbf{x}_i$ in the $[[\mathbf{x}_{\text{pack}}]]$ is the i-th channel data of input ciphertext, and $[\mathbf{MP}_i]$ are the weight plaintext vectors for channel packing defined as $[\mathbf{MP}_i] = ([\mathbf{mp}_0^i], [\mathbf{mp}_1^i], \ldots, [\mathbf{mp}_{k_h k_w - 1}^i])$. $\mathbf{mp}_j^i$ consist of the vectors generated by Algorithm 1. More specifically, $\mathbf{mp}_j^i \in \mathbb{R}^{N/2}$ is defined as:

$$\mathbf{mp}_j^i = \mathsf{concat}(\mathbf{m}_j^{-i,0}, \mathbf{m}_j^{1-i,1}, \ldots, \mathbf{m}_j^{c_n-1-i,c_n-1}), \tag{11}$$

where $j \in [0, k_h k_w - 1]$, $\mathbf{m}_j^{m_o,m_i}$ are $\mathbf{m}_j$ for the m_i-th input channel and the m_o-th output channel, and concat extracts the first $n_{ih} \times n_{iw}$ slots from input vectors $\mathbf{m}_j^{m_o,m_i}$ and concatenates them. When m_o and m_i are negative, c_n is added to set the appropriate channel index. Then, the homomorphic convolution method using channel packing is defined as:

$$[[\hat{\mathbf{y}}_{\text{pack}}]] = \sum_{i=0}^{c_n-1} rot([[\mathbf{x}_{\text{pack}}]] \circledast [\mathbf{MP}_i], n_{ih} n_{iw} \cdot i). \tag{12}$$

Fig. 8 and Eq. (12) show the case of c_i and c_o are the same as c_n. But, these can be extended to support multiple input or output ciphertexts when $c_o > c_n$ or $c_i > c_n$. Table 1 shows the number of rotations and multiplications with and without channel packing. The number of kernel-level rotations is reduced to $(k_h \cdot k_w - 1) \cdot \frac{c_i}{c_n}$ from $(k_h \cdot k_w - 1) \cdot c_i$, as the number of input ciphertexts is reduced to $\frac{c_i}{c_n}$. Similarly, the number of multiplications is reduced to $k_h \cdot k_w \cdot \frac{c_i}{c_n} \cdot c_o$ from $k_h \cdot k_w \cdot c_i \cdot c_o$. Channel-level rotation is performed $(c_n - 1) \cdot \frac{c_o}{c_n}$ times additionally. Additional channel-level rotations are required, but the overall number of operations is significantly reduced, resulting in decreased latency through the channel packing method. Further, this table shows the complexity of multiplexed packing proposed in [15], which reduces the number of multiplications in the convolutional layer by utilizing the non-valid slots. Here, we provide a brief description only of its computational complexity. More detailed explanations can be found in [15]. It packs the ciphertexts by filling the non-valid slots with valid slots from other ciphertexts through rotation and addition. Multiplexed packing reduces the number of kernel-level rotations to $(k_h \cdot k_w - 1) \cdot \frac{c_i}{c_n \cdot g^2}$ from $(k_h \cdot k_w - 1) \cdot \frac{c_i}{c_n}$ and the number of multiplications to $k_h \cdot k_w \cdot \frac{c_i}{c_n \cdot g^2} \cdot c_o$ from $k_h \cdot k_w \cdot \frac{c_i}{c_n} \cdot c_o$.

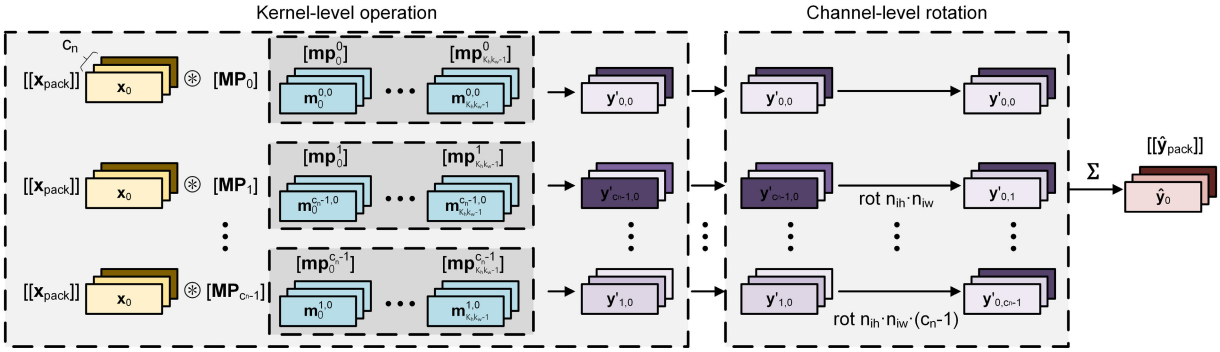

FIGURE 8. Homomorphic convolution method using channel-packed ciphertext. Kernel-level operations are performed, followed by channel-level rotations to sum the intermediate results.

TABLE 2. The network architectures of PP-DNNs.

Network	CNN structure
VGG-16	((C+H)×2 + P)×2 + ((C+H)×3 + P)×3 + (FC +H)×2 + FC
ResNet-18	(C+H)×17 + P + FC
PA-ResNet-18	C+H + (H+C)×16 + P + FC
ResNet-34	(C+H)×33 + P + FC
PA-ResNet-34	C+H + (H+C)×32 + P + FC
ResNet-32	(C+H)×31 + P + FC
PA-ResNet-32	C+H + (H+C)×30 + P + FC

TABLE 3. The accuracy of networks for each dataset.

Dataset	Network	Accuracy (%)	
		ReLU	HerPN [32]
CIFAR-100	VGG-16	72.46	67.18
	ResNet-18	75.66	74.65
	PA-ResNet-18	75.18	74.59
	ResNet-34	77.32	77.35
	PA-ResNet-34	76.41	76.14
CIFAR-10	ResNet-32	93.79	93.12
	PA-ResNet-32	92.81	92.68
Fashion-MNIST	ResNet-32	93.98	93.97
	PA-ResNet-32	94	93.67

However, this method occurs additional rotations during the packing process. Specifically, $\lceil \frac{c_i}{c_n \cdot g^2} \rceil \cdot (g^2 - 1)$ rotations are added for multiplexed packing, and $\lceil \frac{c_o}{c_n} \rceil \cdot \lceil \log(g^2) \rceil$ rotations are required to sum up the multiplexed packed data. In Section IV, we compare this method to the proposed work in more detail.

IV. EVALUATION

A. EXPERIMENTAL SETUP

1) DATASET AND NETWORKS

We evaluate our proposed methods using the Fashion-MNIST [38], CIFAR-10, and CIFAR-100 [39] datasets. The Fashion-MNIST dataset contains 60k train and 10k validation 28 × 28 grayscale images of 10 classes. The CIFAR-10 and CIFAR-100 datasets consist of 60k color images. The training set comprises 50K images, and the test set contains 10K images. The images in these datasets have 3 channels, and each channel is composed of 32 × 32 pixels. The CIFAR-10 has 10 classes, and the CIFAR-100 has 100 classes. The CIFAR-100 dataset is the most complex dataset used in previous PP-DNNs [15], [22], [32].

We adopt several well-known CNN architectures to implement PP-DNNs, as shown in Table 2. C, H, P, and FC denote convolution, HerPN block with three bases, average pooling, and fully-connected layers. We keep the network hyperparameters, such as channel and kernel sizes, the same as those specified in the original papers [1], [2], [3]. All kernels have a size of 3 × 3. In the case of VGG-16, ResNet-18, PA-ResNet-18, ResNet-34, and PA-ResNet-34, the channel size doubles for specific layers, starting at 64 and then increasing to 128, 256, and 512, while the width and height of the image start at 32 × 32 and are halved at each step. In the case of ResNet-32 and PA-ResNet-32, the channel size increases to 16, 32, and 64. We replace the ReLU and batch normalization layers with the HerPN block, as shown in Fig. 4, and replace the max pooling layer with the average pooling layer. We perform experiments on Fashion-MNIST and CIFAR-10 datasets using ResNet-32 [2] and PA-ResNet-32 [3]. For the CIFAR-100 dataset, we employ the VGG-16 [1], ResNet-18 [2], and PA-ResNet-18 [3] used in the previous work [32], and additionally, we evaluate the proposed methods on ResNet-34 and PA-ResNet-34.

The ReLU-based networks and networks with HerPN block are trained by PyTorch in the plaintext domain. We basically follow the training setting of [40] and use the same training settings for all datasets. Images in each dataset are normalized using channel-wise mean and standard deviation for each dataset. Images are zero-padded with an additional 4 pixels around the edges, followed by a random extraction of a 32×32 crop. Images are randomly rotated around, ranging from −15 to +15 degrees, and any resulting blank spaces are filled with zeros and randomly flipped

horizontally with 50% probability. We use He initialization [41] for weight initialization. We utilize SGD for training with a batch size of 128, Nesterov momentum set at 0.9, a weight decay of 5e-4, and employ a cross-entropy loss function. Training is performed in 200 epochs, starting with a learning rate of 0.1, which is reduced by a factor of 5 after 60, 120, and 160 epochs. To train ResNet-32 and PA-ResNet-32 with the Fashion-MNIST dataset, which consists of grayscale images, the grayscale image is duplicated into three RGB channels. Table 3 shows the accuracy of the ReLU-based and the HerPN block-based networks. Unlike other datasets, in the case of the Fashion-MNIST dataset, which is a simpler dataset, it can be observed that ResNet-32 and PA-ResNet-32 achieve nearly the same accuracy. The accuracy of the ReLU-based and HerPN-based networks is almost the same across various networks and datasets. These results show that we can replace the ReLU with the HerPN block while keeping accuracy.

2) SYSTEM SETUP

The inference of all PP-DNNs is executed on a server-scale platform, which consists of an Intel Xeon Gold 6230 at 2.10GHz CPU (20 cores) with 512GB DRAM memory, with the Ubuntu 20.04 operating system. The ciphertext inference is implemented using the RNS-CKKS scheme from the Microsoft SEAL library [42] and the latency is measured using the Chrono C++ library. The scale of the encrypted input message is set to 25 bits, following the approach of [22], and the scale of the weight filters is set to 20 bits, greater than [22] because we have experimented on the deeper network.

B. CKKS PARAMETER SELECTION

To evaluate the effectiveness of the proposed optimization solution for PP-DNN, we implement the PP-DNN operation of previous works and proposed work using the networks described in Table 2. Table 4 shows the setting of the HE parameters, which are systematically selected, for the PP-DNN models used in this work. The HE parameters of this table are configured to enable PP-DNN inference without bootstrapping and all parameters are set to ensure a bit security level λ is greater than or equal to 128-bit security [20], [21], [22]. We set the special prime $\log P = 60$ bits and $q_0 = 60$ bits for all parameters. When using a high-degree polynomial as the activation, the required $\log PQ$ and l_{MAX} increase, necessitating bootstrapping. We successfully eliminate such bootstrapping by using a second-order polynomial as the activation. Further, even when using the same network, the proposed solution, compared to the conventional merging method, reduces the multiplication depth of the convolution and HerPN block pairs from 3 to 2, approximately 33% reduction, through coefficient absorbing and masking convolution. We can also observe a similar trend in both $\log PQ$ and l_{MAX}, which are related to multiplicative depth, in this table. As a result, it can be assumed that the proposed work leads to faster ciphertext inference due to lower values of l_{MAX} and $\log PQ$ compared to the conventional method. In the cases of ResNet-32 and PA-ResNet-32, the proposed work allows to choice of a lower N for the same bit security level due to the decrease in l_{MAX} and $\log PQ$, enabling us to reduce latency even further.

C. RESULTS AND ANALYSIS

To evaluate the effectiveness of the proposed methods, we measure the end-to-end single-image inference latency of PP-DNNs. Table 5 shows the comparison result of inference latency between HEMET [22], MultParConv [15], the proposed work, and the vector domain implemented in PyTorch. Since plaintext in CKKS refers to an encoded polynomial, we use the term vector instead of plaintext to represent the unencoded vector to avoid confusion. We select HEMET and MultParConv as our comparison targets, which use the conventional merging method. We implement HEMET and MultParConv with HerPN block-based networks, as presented in Table 2, for a fair comparison. Both use the conventional merging method, so we apply the conventional parameters from Table 4. Furthermore, for the operation of convolutional layers, HEMET employs channel packing, while MultParConv utilizes multiplexed packing. The proposed solution utilizes the proposed parameters from Table 4 and employs the channel packing method. The latency of VGG-16, ResNet-18, PA-ResNet-18, ResNet-34, and PA-ResNet-34 is measured on the CIFAR-100 dataset, and the latency of ResNet-32 and PA-ResNet-32 is measured on the Fashion-MNIST and CIFAR-10 datasets. To perform inference on Fashion-MNIST, similar to the training process, we replicate the grayscale image into 3 channels and add zero padding to make 32×32 from the 28×28 image. Therefore, as the inference latency of ResNet-32 and PA-ResNet-32 is the same for CIFAR-10 and Fashion-MNIST, we do not include a separate entry for Fashion-MNIST in the table. The results show that the proposed work significantly reduces latency across all networks compared to the HEMET. The latency reduction comes from the decrease in l_{MAX}, which reduces the complexity of PP-DNN defined as $\mathcal{O}(N \cdot \log N \cdot l_{\mathrm{MAX}}^2)$. Specifically, the proposed work reduces the latency by 41% for VGG-16. For ResNet-18,34, PA-ResNet-18, and 34 cases, the proposed work achieves latency reduction by 47%, 59%, 41%, and 58%, respectively. Next, we compare the proposed work with MultParConv, which effectively reduces the total multiplications using multiplexed packing. MultParConv achieves better latency results than HEMET by significantly reducing the number of multiplications. It also achieves better latency than the proposed work on relatively shallow networks. However, as the network layers become deeper, the difference in l_{MAX} between the two methods increases, showing that the proposed work performs better than MultParConv. Specifically, the proposed work has longer latency in VGG-16, ResNet-18, and PA-ResNet-18 compared to MultParConv. However, the proposed work reduces latency by 35% and 31% for ResNet-34 and PA-ResNet-34, respectively, compared to MultParConv. Moreover, in the case of ResNet-32 and

TABLE 4. HE parameter settings for conventional and proposed approaches in this work.

Network	$\log PQ$		$l_{\mathrm{MAX}}+1$		N		λ
	Conventional [15], [22]	Proposed	Conventional [15], [22]	Proposed	Conventional [15], [22]	Proposed	
VGG-16	1330	1060	26	19	2^{16}	2^{16}	≥ 128
ResNet-18	1445	1000	27	19	2^{16}	2^{16}	
PA-ResNet-18	1433	1000	27	19	2^{16}	2^{16}	
ResNet-34	2640	1811	51	35	2^{17}	2^{17}	
PA-ResNet-34	2645	1804	51	35	2^{17}	2^{17}	
ResNet-32	2489	1710	48	33	2^{17}	2^{16}	
PA-ResNet-32	2495	1702	48	33	2^{17}	2^{16}	

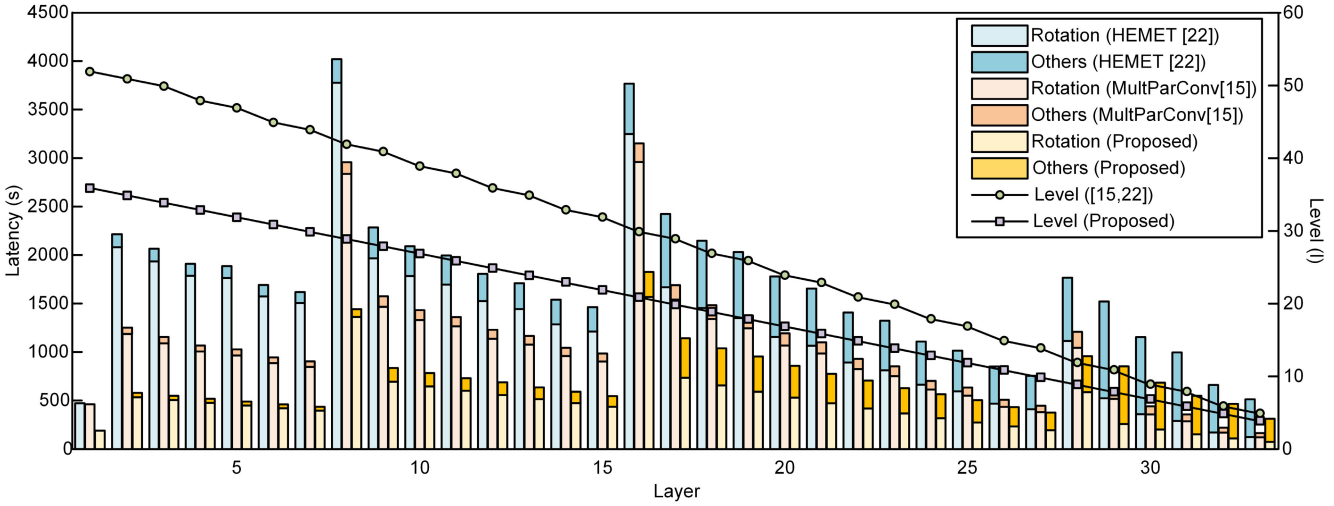

FIGURE 9. Comparison of layer-wise latency in ResNet-34 on CIFAR-100 for the proposed and previous methods.

TABLE 5. Comparison of inference latency between proposed and previous methods.

Network	Latency (s)			
	HEMET [22]	MultParConv [15]	Proposed	PyTorch*
VGG-16	11291	4811	6645	0.013
ResNet-18	6774	3454	3589	0.018
PA-ResNet-18	5902	2995	3481	0.02
ResNet-34	56154	35998	23300	0.035
PA-ResNet-34	54808	33482	23199	0.036
ResNet-32	12081	7469	1749	0.017
PA-ResNet-32	11117	6848	1753	0.017

*Measured results in the vector domain

PA-ResNet-32, as N reduced from 2^{17} to 2^{16}, we can show that the latency is drastically reduced by 77% and 74%, respectively, compared to MultParConv. Then, we compare the inference latency in the HE domain and the vector domain implemented by PyTorch. We use the time library in Python3 to measure the latency of the network implemented in PyTorch running on the CPU. Despite significantly reducing latency in the HE domain through the proposed methods, there still remains a considerable gap when compared to the latency in the vector domain. It can be observed that compared to the latency in the vector domain, the latency in the HE domain is at least 100,000 times slower.

Then, we analyze the effect of the proposed methods in terms of layer-wise latency of the network. Fig. 9 shows the layer-wise latency analysis of convolutional layers in ResNet-34 for the HEMET, MultParConv, and the proposed methods. Others include plaintext multiplication and addition latency. We have not analyzed the latency of activation functions (HerPN block) and rescaling operations as it is small enough to be considered negligible. Due to each ciphertext having $\lfloor \frac{N}{2\cdot(W+2\cdot P)\cdot(H+2\cdot P)} \rfloor$ channels in HEMET [20], [22], this work has more input and output ciphertexts of each layer than MultParConv and the proposed work. Furthermore, as mentioned earlier, since the coefficient of the highest term of the activation function is not absorbed, requiring a more multiplicative depth, the l_{MAX} is large. On the other hand, since both MultParConv and

TABLE 6. Implementation results of the previous and proposed methods with Lattigo and SEAL for ResNet-32 on CIFAR-10.

Library	Latency(s)		
	HEMET [22]	PrlMultConv [15]	Proposed
SEAL	12081	7469	1749
Lattigo	8175	4967	1212

the proposed work pack the $\lfloor \frac{N}{2 \cdot W \cdot H} \rfloor$ channels into each ciphertext, it is obvious that the number of ciphertexts is lower than HEMET. Also, MultParConv significantly reduces the number of multiplications through multiplexed packing, greatly decreasing the ratio of other operations in the total latency compared to other works. However, MultParConv has limitations in latency reduction due to the required larger l for mask operations on non-valid slots. As the latency of rotation and plaintext multiplication is proportional to l^2 and l, as shown in Fig. 2, there is a substantial difference in latency between rotation and multiplication for a large l. Therefore multiplexed packing effectively reduces the overall multiplication but increases the level l, making the rotation latency too long. On the other hand, the proposed work, which further reduces the level l, effectively reduces the overall latency in deep neural networks, particularly in the earlier layers of the network. It almost reduces 50% latency of earlier layers compared to the MultParConv. The proposed work suggests that its effectiveness increases as the total required multiplication depth of PP-DNN grows larger.

Lastly, we present the implementation results of the previous and proposed methods using a different HE library in Table 6. The implementation in a different HE library is conducted for ResNet-32 on CIFAR-10. We use the Lattigo [43] as the alternative HE library for the implementation. To measure the latency of the network implemented with Lattigo, we use the time library in the Go language. Our optimization solution can be confirmed to work well in Lattigo as well. As presented in [14], [44], Lattigo performs optimization at various levels, resulting in at least 1.44× speed improvement compared to SEAL when running the same algorithm.

D. DISCUSSION

It is important to reduce level l to reduce latency as the latency of HE operation is related to l. In this work, to further reduce the level l while preserving accuracy, we adopt degree-2 polynomial as non-linear activation and propose multiplicative depth reduction techniques called coefficient absorbing and masking convolution. The proposed methods reduce the multiplication depth by 33% compared to the previous merging method, and as a result, both $\log PQ$ and l_{MAX}, which have a strong correlation with the multiplication depth, in Table 4 are similarly reduced. Therefore, through the proposed method, it is possible to implement PP-DNN with feasible HE parameters without bootstrapping, as presented in Table 4, even for deep networks.

Through latency comparison with previous approaches, we show the validity of the proposed optimization solution. As shown in Table 5, compared to HEMET [22], our approaches achieve latency reductions of 1.7 to 6.9 times in various networks. Compared to MultParConv [15], the proposed approaches reduce latency by 1.4 to 3.9 times in deep networks. However, the proposed methods show longer latencies in relatively shallow networks. More specifically, in our implementation, our method shows better performance when $\log PQ$ of MultParConv is higher than 1761, i.e., $\log PQ$ of MultParConv is higher than the 128-bit security level value for $N = 2^{16}$. This is because, as shown in Fig. 2, the rotation latency is proportional to l^2. Therefore, the proposed solution indicates that the larger the total required multiplication depth of PP-DNN, the more effective it is due to the gap in level l between the two methods widening. The proposed work may show lower performance in shallow networks, but in practical applications, using deep networks is common to achieve higher accuracy. Therefore, the proposed work gains advantages by eliminating bootstrapping for deep networks and reducing the multiplicative depth more than the conventional method. Despite our optimization technique, as shown in Table 5, HE domain inference still requires long latency compared to vector domain inference. The implementation results are based solely on CPU execution, and the proposed method is not platform-specific optimization, so it is anticipated that utilizing GPUs will result in a reduction of latency by over 100 times [45]. However, comparing latencies based on different computing platforms falls outside the scope of our current work, so we leave it for future work.

Through layer-wise latency analysis in Fig. 9, it can be observed that the proposed work achieves a significant reduction in latency at earlier layers of ResNet-34 compared to conventional merging method-based works, specifically, a 3.8× and 2× reduction in latency compared to HEMET and MultParConv, respectively.

In addition, we show that the proposed work can be applied to various scenarios through various generality evaluations, such as datasets, networks, and HE libraries. Therefore, the proposed work can be applied to all convolution-based networks, making it applicable for various image tasks such as medical, satellite, and facial image processing. However, the current proposed work is centered on CNN optimization techniques, and the previous works [15], [22], [37] are also confined to CNNs. Therefore, applying this approach to LSTM or non-convolution-based ANN would require further research and is left for future work.

V. CONCLUSION

In this paper, we have proposed a low-latency model optimization solution for non-interactive PP-DNN based

on the RNS-CKKS scheme. We utilize the HerPN block as an activation function to achieve low-latency inference while maintaining the accuracy of the original model. To further reduce the multiplicative depth compared to the previous works, coefficient absorbing and masking convolution schemes are proposed. The proposed approaches can construct the HE-aware convolution layers without affecting the recognition result. As a result, the proposed work constructs the PP-DNN without the need for bootstrapping while preserving the baseline accuracy using practical HE parameters. Compared with the state-of-the-art, the proposed work reduces the processing latency of ResNet-32 by 4.3 times on the Fashion-MNIST and CIFAR-10 datasets and ResNet-34 by 1.5 times on the CIFAR-100 dataset. Further, through layer-wise latency analysis, we show the effectiveness of the proposed work in the deep network, especially. The evaluation results using various datasets, networks, and HE libraries demonstrate that the proposed solution can be generally applicable. Thus, our methods provide a practical and efficient solution for accelerating secure deep neural network computations in a privacy-preserving manner. In future work, our objective is to address more challenging tasks about PP-DNNs. This includes exploring solutions such as GPU or dedicated hardware processing, additional optimization techniques for various architectures, and applications to deep network architectures.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2016, pp. 630–645.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, and S. Gelly, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*, 2020.

[5] H. Lee, Y.-S. Kim, M. Kim, and Y. Lee, "Low-cost network scheduling of 3D-CNN processing for embedded action recognition," *IEEE Access*, vol. 9, pp. 83901–83912, 2021.

[6] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[7] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6450–6459.

[8] J. Jo, J. Kung, S. Lee, and Y. Lee, "Similarity-based LSTM architecture for energy-efficient edge-level speech recognition," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.

[9] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5884–5888.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. Askell, "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

[11] 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*.

[12] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via MiniONN transformations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 619–631.

[13] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," *BMC Med. Genomics*, vol. 11, no. S4, pp. 23–31, Oct. 2018.

[14] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," in *Proc. EUROCRYPT*. Cham, Switzerland: Springer, 2021, pp. 587–617.

[15] E. Lee, J.-W. Lee, J. Lee, Y.-S. Kim, Y. Kim, J.-S. No, and W. Choi, "Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 12403–12422.

[16] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. ASIACRYPT*, 2017, pp. 409–437.

[17] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Proc. EUROCRYPT*. Cham, Switzerland: Springer, 2018, pp. 360–384.

[18] 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.*, 2016, pp. 201–210.

[19] A. Brutzkus, R. Gilad-Bachrach, and O. Elisha, "Low latency privacy preserving inference," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 812–821.

[20] 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.*, Jun. 2019, pp. 142–156.

[21] R. Dathathri, B. Kostova, O. Saarikivi, W. Dai, K. Laine, and M. Musuvathi, "EVA: An encrypted vector arithmetic language and compiler for efficient homomorphic computation," in *Proc. 41st ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Jun. 2020, pp. 546–561.

[22] Q. Lou and L. Jiang, "HEMET: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 7102–7110.

[23] M. Kim, X. Jiang, K. Lauter, E. Ismayilzada, and S. Shams, "Secure human action recognition by encrypted neural network inference," 2021, *arXiv:2104.09164*.

[24] 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.

[25] T. Xie, H. Yamana, and T. Mori, "CHE: Channel-wise homomorphic encryption for ciphertext inference in convolutional neural network," *IEEE Access*, vol. 10, pp. 107446–107458, 2022.

[26] J. Qian, P. Zhang, H. Zhu, M. Liu, J. Wang, and X. Ma, "LHDNN: Maintaining high precision and low latency inference of deep neural networks on encrypted data," *Appl. Sci.*, vol. 13, no. 8, p. 4815, Apr. 2023.

[27] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proc. USENIX Secur. Symp.*, 2018, pp. 1651–1669.

[28] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference system for neural networks," in *Proc. Workshop Privacy-Preserving Mach. Learn. Pract.*, Nov. 2020, pp. 2505–2522.

[29] Z. Ghodsi, A. K. Veldanda, B. Reagen, and S. Garg, "CryptoNAS: Private inference on a ReLU budget," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 16961–16971.

[30] N. K. Jha, Z. Ghodsi, S. Garg, and B. Reagen, "DeepReDuce: ReLU reduction for fast private inference," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 4839–4849.

[31] Q. Lou, Y. Shen, H. Jin, and L. Jiang, "SAFENet: A secure, accurate and fast neural network inference," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–13.

[32] J. Park, M. J. Kim, W. Jung, and J. H. Ahn, "AESPA: Accuracy preserving low-degree polynomial activation for fast private inference," 2022, *arXiv:2201.06699*.

[33] 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," *IEEE Access*, vol. 11, pp. 62062–62076, 2023.
[34] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Proc. CRYPTO*, 2012, pp. 868–886.
[35] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *Proc. Int. Conf. Sel. Areas Cryptogr.*, 2018, pp. 347–368.
[36] R. Ge, J. D. Lee, and T. Ma, "Learning one-hidden-layer neural networks with landscape design," 2017, *arXiv:1711.00501*.
[37] V. S. Lokhande, S. Tasneeyapant, A. Venkatesh, S. N. Ravi, and V. Singh, "Generating accurate pseudo-labels in semi-supervised learning and avoiding overconfident predictions via Hermite polynomial activations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11432–11440.
[38] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[39] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[40] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[42] *Microsoft SEAL (Release 3.7)*, Microsoft Research, Redmond, WA, USA, Sep. 2021. [Online]. Available: https://github.com/Microsoft/SEAL
[43] (Aug. 2022). *Lattigo v4*. ePFL-LDS, Tune Insight SA. [Online]. Available: https://github.com/tuneinsight/lattigo
[44] C. V. Mouchet, J.-P. Bossuat, J. R. Troncoso-Pastoriza, and J.-P. Hubaux, "Lattigo: A multiparty homomorphic encryption library in go," in *Proc. 8th Workshop Encrypted Comput. Appl. Homomorphic Cryptogr.*, 2020, pp. 64–70.
[45] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, "Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with GPUs," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 114–148, Aug. 2021.

HYUNHOON LEE (Graduate Student Member, IEEE) received the B.S. degree in electronics from Kyungpook National University, Daegu, South Korea, in 2018, and the M.S. degree in electrical engineering from the Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 2020, where he is currently pursuing the Ph.D. degree.

His current research interests include homomorphic encryption, privacy-preserving machine learning, and architectures for digital systems.

YOUNGJOO LEE (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2008, 2010, and 2014, respectively.

Since 2017, he has been with the Department of Electrical Engineering, Pohang University of Science and Technology (POSTECH), Pohang, South Korea, where he is currently an Associate Professor. Prior to joining POSTECH, he was with the Interuniversity Microelectronics Center (IMEC), Leuven, Belgium, from 2014 to 2015, where he researched reconfigurable SoC platforms for software-defined radio systems. From 2015 to 2017, he was with the Department of Electronic Engineering, Kwangwoon University, Seoul, South Korea. His current research interests include algorithms and architectures for embedded processors, intelligent multimedia systems, advanced error-correction codes, and next-generation wireless systems.