IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 May 2023, accepted 15 June 2023, date of publication 19 June 2023, date of current version 23 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3287564

APPLIED RESEARCH

Precise Approximation of Convolutional Neural Networks for Homomorphically Encrypted Data

JUNGHYUN LEE[1], (Graduate Student Member, IEEE), EUNSANG LEE[2], JOON-WOO LEE[3], (Member, IEEE), YONGJUNE KIM[4], (Member, IEEE), YOUNG-SIK KIM[5], (Member, IEEE), AND JONG-SEON NO[1], (Fellow, IEEE)
[1]Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, Republic of Korea
[2]Department of Software, Sejong University, Seoul 05006, Republic of Korea
[3]Department of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea
[4]Department of Electrical Engineering, Pohang University of Science and Technology, Pohang 37673, Republic of Korea
[5]Department of Information and Communication Engineering, Chosun University, Gwangju 61452, Republic of Korea

Corresponding authors: Eunsang Lee (eslee3209@sejong.ac.kr) and Joon-Woo Lee (jwlee2815@cau.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government (MSIT, Ministry of Science and ICT) under Grant NRF-2021R1A2C2011082, and in part by the NRF grant funded by the MSIT (No. RS-2023-00212103).

ABSTRACT Homomorphic encryption (HE) is one of the representative solutions to privacy-preserving machine learning (PPML) classification enabling the server to classify private data of clients while guaranteeing privacy. This work focuses on PPML using word-wise fully homomorphic encryption (FHE). In order to implement deep learning on word-wise HE, the ReLU and max-pooling functions should be approximated by polynomials for homomorphic operations. Most of the previous studies focus on HE-friendly networks, which approximate the ReLU and max-pooling functions using low-degree polynomials. However, this approximation cannot support deeper neural networks due to large approximation errors in general and can classify only relatively small datasets. Thus, we propose a precise polynomial approximation technique, a composition of minimax approximate polynomials of low degrees for the ReLU and max-pooling functions. If we replace the ReLU and max-pooling functions with the proposed approximate polynomials, standard deep learning models such as ResNet and VGGNet can still be used without further modification for PPML on FHE. Even pre-trained parameters can be used without retraining, which makes the proposed method more practical. We approximate the ReLU and max-pooling functions in the ResNet-152 using the composition of minimax approximate polynomials of degrees 15, 27, and 29. Then, we succeed in classifying the plaintext ImageNet dataset with 77.52% accuracy, which is very close to the original model accuracy of 78.31%. Also, we obtain an accuracy of 87.90% for classifying the encrypted CIFAR-10 dataset in the ResNet-20 without any additional training.

INDEX TERMS Fully homomorphic encryption, RNS-CKKS, privacy-preserving machine learning, deep learning, cloud computing.

I. INTRODUCTION

Machine learning as a service (MLaaS) is a service where a client sends data to a server through a cloud service and obtains analysis results after the server performs machine learning. With the innovative development of deep learning technology, the demand for MLaaS is rapidly increasing. However, some data such as medical or financial data is very sensitive, and thus the privacy-preserving technique is indispensable for these data to be used in MLaaS. Homomorphic encryption (HE) is one of the most promising solutions for MLaaS that preserves privacy, called privacy-preserving machine learning (PPML). Using HE, the server can perform deep learning on the encrypted data from the client while fundamentally preventing the privacy leakage of private data. Thus, the implementation of PPML using HE has been actively studied [1], [2], [3], [4], [5], [6], [7].

A fully homomorphic encryption (FHE) technique that supports algebraic operations without the restriction on the number of operations has been developed, but should be

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

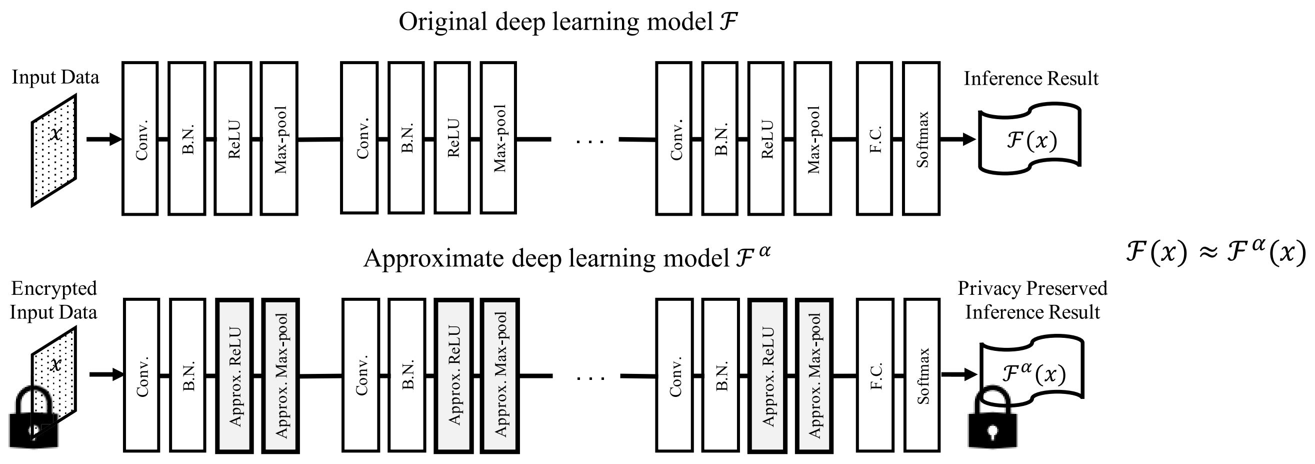

Original deep learning model $\mathcal{F}$
Input Data
x
Conv.
B.N.
ReLU
Max-pool
...
F.C.
Softmax
Inference Result
$\mathcal{F}(x)$
Approximate deep learning model $\mathcal{F}^{\alpha}$
$\mathcal{F}(x) \approx \mathcal{F}^{\alpha}(x)$
Encrypted Input Data
Approx. ReLU
Approx. Max-pool
Privacy Preserved Inference Result
$\mathcal{F}^{\alpha}(x)$

FIGURE 1. **Comparison between the original deep learning model $\mathcal{F}$ and the proposed approximate deep learning model $\mathcal{F}^{\alpha}$ with the precision parameter α. The proposed model is obtained by replacing the ReLU and max-pooling functions of the original models with the proposed approximate polynomials.**

addressed to realize practical FHE techniques. The high-latency of FHE operations and large bootstrapping errors have been pointed out as the reason for the low practicality of the FHE technique. Thus, recent works focus on HE-friendly networks in leveled HE. Usually, HE-friendly networks have limited layers and use low-degree approximate polynomials for activation functions and max-pooling. However, this approach has the critical limitations as follows. First, HE-friendly networks have not yet achieved high classification accuracy for large datasets such as ImageNet [8] although they succeeded in classifying small datasets, e.g., MNIST and CIFAR-10 [9]. Secondly, for HE-friendly networks, pre-trained parameters for standard neural networks cannot be used, and thus training is required to optimize the parameters of HE-friendly networks. Since training is a costly and time-consuming process, being able to avoid training is a great advantage. Furthermore, access to training datasets requires prior approval of data owners or some regulatory authorities, and thus, it is not even possible to train for new models in many real-world applications. Therefore, it is a practically important research topic to remove the need for designing a new specialized HE-friendly network and to perform the standard deep neural network on HE without additional training.

It can be an alternative solution to perform standard deep neural networks using multi-party computation (MPC) techniques such as Garbled circuits together with HE. Gazelle [1], Cheetah [2], nGraph-HE [3], and Delphi [4] are representative works using the MPC techniques. However, the MPC techniques result in huge communication costs especially for standard deep neural networks that require a number of activation functions. According to Delphi [4], an exact ReLU operation using Garbled circuit requires a communication cost of about 19kB, and thus several gigabytes of communication cost per image is required to perform standard deep neural network such as ResNet-20 or ResNet-32. Thus, this MPC approach would be impractical for applications that cannot support sufficient communication resources.

Fortunately, FHE techniques have recently been improved a lot both in terms of computation time and bootstrapping accuracy [10], [11], [12], [13], so we can expect to perform standard deep neural networks with small communication costs using only FHE. Since FHE supports arbitrarily large depths of operations, it can support very deep neural networks, i.e., neural networks with many layers. However, non-arithmetic operations such as ReLU and max-pooling are not supported by word-wise FHE. Hence, it is important to handle these non-arithmetic operations using word-wise FHE.

A. OUR CONTRIBUTIONS

1) PROPOSITION OF PRECISE APPROXIMATION BASED ON THE COMPOSITION OF MINIMAX APPROXIMATE POLYNOMIALS

In this paper, we propose to replace the ReLU and max-pooling functions with precise approximate polynomials so that any standard networks can be performed using pre-trained parameters (trained for standard network) without the need for designing a new HE-friendly model and additional training. One might argue that this can be achieved using just high-degree Taylor approximate polynomials or minimax approximate polynomials. However, according to our analysis in Section III-C, precise approximation using a single polynomial requires a polynomial of degree at least about $2^{1.0013\alpha-2.8483}$ for the precision parameter α, which results in $2^{\Theta(\alpha)}$ number of non-scalar multiplications and several numerical issues, where the notation Θ represents computational complexity. Thus, we propose a precise polynomial approximation technique for the ReLU and max-pooling functions that uses a composition of minimax approximate polynomials of low degrees. We replace the

ReLU and max-pooling functions with the proposed approximate polynomials in well-studied deep learning models such as ResNet [14], VGGNet [15], and GoogLeNet [16], and the pre-trained parameters can be used without retraining. Fig. 1 shows the comparison between the original deep learning model $\mathcal{F}$ and the proposed approximate deep learning model $\mathcal{F}^{\alpha}$ for the precision parameter α, where $\mathcal{F}$ is the function that outputs the inference result for an input $\mathbf{x}$.

2) THEORETIC ANALYSIS OF THE PROPOSED APPROXIMATE DEEP LEARNING MODEL

In addition, we theoretically analyze the performance of our proposed precise polynomial approximation technique. In Section IV, we prove that the two inference results, $\mathcal{F}^{\alpha}(\mathbf{x})$ and $\mathcal{F}(\mathbf{x})$, satisfy

$$||\mathcal{F}^{\alpha}(\mathbf{x}) - \mathcal{F}(\mathbf{x})||_{\infty} \leq C \cdot 2^{-\alpha} \tag{1}$$

for a given precision parameter α and a constant C, which can be determined independently of the precision parameter α (see Theorem 4). This implies if we increase the precision parameter α enough, the two inference results would be almost identical.

3) NUMERICAL ANALYSIS OF THE PROPOSED APPROXIMATE DEEP LEARNING MODEL

Finally, we provide simulation results of the proposed method for standard deep learning models and real data, which supports our theoretical analysis. We classify the CIFAR-10 without retraining using approximate well-studied deep learning models such as ResNet and VGGNet. For classifying the ImageNet in ResNet-152, we approximate the ReLU and max-pooling functions using the composition of minimax approximate polynomials of degrees 15, 27, and 29. Then, for the first time for PPML using word-wise HE, we successfully classify the ImageNet with 77.52% accuracy, which is very close to the original model accuracy of 78.31%. The proposed approximate polynomials are generally applicable to various deep learning models such as ResNet, VGGNet, and GoogLeNet for PPML on FHE. Although some simulations are performed for plaintext data, they can also be performed for ciphertext since all non-arithmetic operations have been replaced with polynomial operations. To show the validity of our results for plaintext data, we also provide the results of a classification in ResNet-20 and ResNet-32 for the CIFAR-10 dataset on the encrypted data. These show that the classification accuracy results for plaintext data are very close to those for encrypted data.

B. RELATED WORKS

PPML using HE can be classified into PPML using bit-wise HEs and PPML on word-wise HEs. PPML using the fast fully homomorphic encryption over the torus (TFHE) [23], which is one of the most promising bit-wise HEs, was studied [24], [25], [26], [27]. This PPML has advantages in that it can evaluate some non-arithmetic operations, such as the ReLU and max-pooling functions exactly on the encrypted data. However, word-wise HEs such as Cheon-Kim-Kim-Song (CKKS) [28] scheme have a great advantage in terms of computation time because they can perform many operations of real numbers or integers simultaneously using packing. In this paper, we focus on PPML using word-wise HE.

So far, most studies using word-wise HEs require HE-friendly networks to perform classification with small depth consumption. HE-friendly networks use low-degree polynomials to replace non-arithmetic activation functions. For instance, CryptoNets [5], SEALion [6], HCNN [17] and BAYHENN [18] used x^2 as an activation function, and Faster CryptoNets [7] used a quantized approximate ReLU function polynomial (ReLU-AQ) $2^{-3}x^2 + 2^{-1}x + 2^{-2}$. The authors in [19] used polynoimals of degree 2–4, approximated by least square method. CryptoDL [20] used a cubic polynomial, which is an integration of approximate sigmoid function, motivated by the similarity between the sigmoid function and the derivative of the ReLU function. TensorHE [21] used a similar method as CryptoDL, integrating the approximate step function instead of the sigmoid function. CHET [22] used $ax^2 + bx$ as an activation function, where a and b are learnable parameters that can be obtained in a training phase. Since these HE-friendly networks do not allow the use of pre-trained parameters for standard networks, they require the whole process of training. Table 1 shows the comparison between the previous PPML using word-wise HEs and the proposed method.

TABLE 1. Comparison Between Previous Word-wise HE PPML and the Proposed Method.

Datasets	HE-friendly networks	Pre-trained standard networks
MNIST	CryptoNets [5] SEALion [6] HCNN [17] BAYHENN [18] FasterCryptoNets [7] Least square [19] CryptoDL [20] TensorHE [21] CHET [22]	-
CIFAR-10	HCNN [17] FasterCryptoNets [7] Least square [19] CryptoDL [20] TensorHE [21] CHET [22]	Approximate CNN (Proposed)
ImageNet	*Not reported*	Approximate CNN (Proposed)

Recently, with the progress in efficient computation techniques for FHE scheme, the technological advancement in implementing deep neural networks has made it possible not only for simple models like HE-friendly networks but also for deeper networks that achieve high performance. The authors in [29] successfully classified encrypted CIFAR-10 data for the first time by implementing ResNet-20 using the residue number system variant CKKS (RNS-CKKS) scheme [30]. Also, the authors in [31] used multiplexed parallel convolution to minimize boostrapping runtime and imaginary-removing boortrapping to reduce the error while evaluating deep circuit. In this paper, we leverage this implementation to evaluate the proposed precise approximation of standard neural networks. This paper does not include the details of such implementation methods as in [29] and [31], but focuses on approximating non-arithmetical operations to polynomials.

II. PRELIMINARIES

A. RNS-CKKS SCHEME

To preserve the privacy of data, one can encrypt the data with a cryptographic system before sending the data to the server. If we use the FHE scheme as a cryptographic system, some homomorphic operations are available between ciphertexts without any information leakage of plaintexts. Let us denote $\mathtt{Enc}_k(m)$ as the ciphertext, which encrypts of the message m with secret key k. The RNS-CKKS scheme encrypts m, the vector of real (or complex) numbers, as the pair of polynomials $(a, b) \in \mathcal{R}_Q^2$, where $\mathcal{R}_Q = \mathbb{Z}_Q[X]/\langle X^N + 1\rangle$ for some large integer parameters N and Q. There are homomorphic operations between ciphertexts to perform arithmetic operations without decrypting any ciphertexts. The word-wise FHE scheme supports the homomorphic addition $\oplus$ and homomorphic multiplication $\otimes$, which satisfies:

- $\mathtt{Enc}_k(m_1) \oplus \mathtt{Enc}_k(m_2) = \mathtt{Enc}_k(m_1 + m_2)$,
- $\mathtt{Enc}_k(m_1) \otimes \mathtt{Enc}_k(m_2) = \mathtt{Enc}_k(m_1 \times m_2)$,

where $m_1 + m_2$ and $m_1 \times m_2$ denote the componentwise vector addition and multiplication between plantexts, respectively. The detailed techniques about these homomorphic operations are described in [28] and [30].

B. DEEP LEARNING ON FULLY HOMOMORPHIC ENCRYPTION

This subsection illustrates the progress of deep learning inference for fully homomorphically encrypted data.

Let $\mathbf{x}$ denote the input data that the client owns, and $\mathcal{F}$ denote the deep learning model circuit owned by the server. The client aims to obtain the inference result $\mathcal{F}(\mathbf{x})$. Due to the privacy issue, the client cannot send $\mathbf{x}$ directly to the server. Therefore, prior to sending the data to the server, the client encrypts the data using the FHE scheme as $\mathtt{Enc}_k(\mathbf{x})$, where k denotes the client's own secret key.

The primary objective of the server is to obtain a ciphertext encrypting $\mathcal{F}(\mathbf{x})$, namely $\mathtt{Enc}_k(\mathcal{F}(\mathbf{x}))$, without decrypting any ciphertexts. If the server can send the ciphertext $\mathtt{Enc}_k(\mathcal{F}(\mathbf{x}))$ to the client, the client can decrypt the ciphertext using the secret key and obtain the inference result $\mathcal{F}(\mathbf{x})$. The server can empoly homomorphic operations, such as $\oplus$ and $\otimes$, to execute the circuit $\mathcal{F}$.

Unfortunately, the server cannot obtain the exact encryption of the inference result $\mathcal{F}(\mathbf{x})$ if the circuit $\mathcal{F}$ includes non-arithmetic operations like ReLU and max-pooling functions. One approach to address this issue is to approximate non-arithmetic operations using polynomials. Let $\mathcal{F}^\alpha$ denote a circuit in which all non-arithmetic operations in $\mathcal{F}$ are substituted with polynomials. Consequently, the server can replace every addition and multiplication in the circuit $\mathcal{F}^\alpha$ with the homomorphic operations $\oplus$ and $\otimes$, respectively. Let $\bar{\mathcal{F}}^\alpha$ be such a circuit. Since $\bar{\mathcal{F}}^\alpha$ is composed solely of homomorphic operations, the following equation holds:

$$\bar{\mathcal{F}}^\alpha(\mathtt{Enc}_k(\mathbf{x})) = \mathtt{Enc}_k(\mathcal{F}^\alpha(\mathbf{x})).$$

Therefore, the decryption of the ciphertext $\bar{\mathcal{F}}^\alpha(\mathtt{Enc}_k(\mathbf{x}))$ is the same as $\mathcal{F}^\alpha(\mathbf{x})$.

In summary, the server approximates the non-arithmetic operations present in the original circuit $\mathcal{F}$ and constructs the circuit $\bar{\mathcal{F}}^\alpha$. Then, the server obtains the ciphertext $\bar{\mathcal{F}}^\alpha(\mathtt{Enc}_k(\mathbf{x}))$ and transmits it to the client. The client decrypts the received ciphertext and get the approximate inference result $\mathcal{F}^\alpha(\mathbf{x})$.

C. MINIMAX COMPOSITE POLYNOMIAL

Since the word-wise FHE schemes do not support non-arithmetic operations such as ReLU and max-pooling, these non-arithmetical functions should be approximated by polynomials. Recently, the authors of [32] proposed a composite polynomial named *minimax composite polynomial* that optimally approximates the sign function with respect to the depth consumption and the number of non-scalar multiplications. The definition of *minimax composite polynomial* is given as follows:

Definition 1: Let D be $[-b, -a] \cup [a, b]$. A composite polynomial $p_k \circ p_{k-1} \circ \cdots \circ p_1$ is called a minimax composite polynomial on D for $\{d_i\}_{1 \le i \le k}$ if the followings are satisfied:

- *p_1 is the minimax approximate polynomial of degree at most d_1 on D for* $\mathrm{sgn}(x)$.
- *For $2 \le i \le k$, p_i is the minimax approximate polynomial of degree at most d_i on $p_{i-1} \circ p_{i-2} \circ \cdots \circ p_1(D)$ for* $\mathrm{sgn}(x)$.

For ϵ $(0 < \epsilon < 1)$ and $\beta > 0$, a polynomial $p(x)$ that approximates $\mathrm{sgn}(x)$ is called (β, ϵ)-close if it satisfies $|p(x) - \mathrm{sgn}(x)| \le 2^{-\beta}$ for $x \in [-1, -\epsilon] \cup [\epsilon, 1]$ [33]. Then, the optimal (β, ϵ)-close composite polynomial $p(x)$ is the minimax composite polynomial on $[-1, -\epsilon] \cup [\epsilon, 1]$ for $\{d_i\}_{1 \le i \le k}$, where $\{d_i\}_{1 \le i \le k}$ is obtained from Algorithm 5 for inputs β and ϵ in [32].

D. HOMOMORPHIC MAX FUNCTION

To perform the max function on the encrypted data, called the homomorphic max function, a polynomial that approximates the max function should be evaluated on the encrypted data instead of the max function. The approximate polynomial of the max function, $m(a, b)$, should satisfy the following inequality for the precision parameter $\alpha > 0$:

$$|m(a, b) - \max(a, b)| \le 2^{-\alpha} \quad \text{for } a, b \in [0, 1]. \tag{2}$$

By considering $\max(a, b) = \frac{(a+b)+(a-b)\,\mathrm{sgn}(a-b)}{2}$, a polynomial $m(a, b)$ that approximates $\max(a, b)$ can be

TABLE 2. The Optimized Max Function Factor, Degrees of the Component Polynomials, and Depth Consumption for Evaluating $m_\alpha(a, b)$ According to the Precision Parameter α [32].

α	ζ_α	degrees	depth
4	5	{5}	4
5	5	{13}	5
6	10	{3,7}	6
7	11	{7,7}	7
8	12	{7,15}	8
9	13	{15,15}	9
10	13	{7,7,13}	11
11	15	{7,7,27}	12
12	15	{7,15,27}	13
13	16	{15,15,27}	14
14	17	{15,27,29}	15

obtained by

$$m(a, b) = \frac{(a + b) + (a - b)p(a - b)}{2}, \tag{3}$$

where $p(x)$ is an approximate polynomial for the sign function [33]. In [32], the authors proposed a polynomial $m(a, b)$ that approximates the max function using a minimax composite polynomial for $p(x)$ and (3) [32].

Specifically, for a precision parameter α and the optimized max function factor $\zeta_\alpha > 0$, an $(\alpha - 1, \zeta_\alpha \cdot 2^{-\alpha})$-close minimax composite polynomial $p_\alpha(x) = p_{\alpha,k} \circ \cdots \circ p_{\alpha,1}(x)$ was obtained by using the optimal degrees obtained from Algorithm 5 in [32], which minimizes the depth consumption. Then, it was shown that $m_\alpha(a, b) = \frac{(a+b)+(a-b)p_\alpha(a-b)}{2}$ from $p_\alpha(x)$ satisfies the error condition in (2). Table 2 lists the values of ζ_α, corresponding depth consumption for evaluating $m_\alpha(a, b)$, and degrees of the component polynomials $p_{\alpha,1}, \cdots, p_{\alpha,k}$ according to α. This approximate polynomial $m_\alpha(a, b)$ has the best performance up to now.

III. PRECISE POLYNOMIAL APPROXIMATION OF THE ReLU AND MAX-POOLING FUNCTIONS

The ReLU and max-pooling functions can be approximated by Taylor polynomials or minimax approximate polynomials [34]. However, these approximate polynomials with minimal approximation errors require many non-scalar multiplications. Hence, we approximate the ReLU and max-pooling functions using $p_\alpha(x)$ and $m_\alpha(a, b)$, which are defined in Section II-D. Although we focus on the ReLU function, the proposed approximation method can be extended to other activation functions such as the leaky ReLU.

A. PRECISE APPROXIMATE POLYNOMIAL OF THE ReLU FUNCTION

In this subsection, we propose a polynomial that precisely approximates the ReLU function, which is the most widely used activation function. For a given precision parameter α, it is required that the approximate polynomial of the ReLU function, $r(x)$, satisfies the following error condition:

$$|r(x) - \mathrm{ReLU}(x)| \le 2^{-\alpha} \quad \text{for } x \in [-1, 1]. \tag{4}$$

Since $\mathrm{ReLU}(x) = \frac{x + x\,\mathrm{sgn}(x)}{2}$, we propose the following approximate polynomial of the ReLU function, $r_\alpha(x) = \frac{x + xp_\alpha(x)}{2}$. Fig. 2 shows the component polynomials of $p_\alpha(x)$ and the proposed $r_\alpha(x)$ with the precision parameter $\alpha = 11$. The following theorem shows that the proposed $r_\alpha(x)$ satisfies the error condition of the ReLU function in (4).

Theorem 1: For any $\alpha > 0$, $r_\alpha(x)$ satisfies the following inequality:

$$|r_\alpha(x) - \mathrm{ReLU}(x)| \le 2^{-\alpha} \quad \textit{for } x \in [-1, 1].$$

Proof: The proof is given in Appendix A. ■

Approximation range. The value of polynomial $r_\alpha(x)$ is close to the ReLU function value only for $x \in [-1, 1]$. Since input values of the ReLU function during deep learning are not limited to $[-1, 1]$, the approximation range should be extended from $[-1, 1]$ to a larger range $[-B, B]$ for some $B > 1$. Thus, we propose a polynomial $\tilde{r}_{\alpha,B}(x) := Br_\alpha(x/B)$ approximating the ReLU function in the range $[-B, B]$. Then, we have

$$\begin{aligned}
&|\tilde{r}_{\alpha,B}(x) - \mathrm{ReLU}(x)| \\
&= |B\tilde{r}_{\alpha,B}(x/B) - \mathrm{ReLU}(x)| \\
&= |B\tilde{r}_{\alpha,B}(x/B) - B\,\mathrm{ReLU}(x/B)| \\
&= |B(\tilde{r}_{\alpha,B}(x/B) - \mathrm{ReLU}(x/B))| \le B2^{-\alpha}
\end{aligned}$$

for $x \in [-B, B]$. There is a trade-off relation between the input range and the approximation error.

B. PRECISE APPROXIMATE POLYNOMIAL OF THE MAX-POOLING FUNCTION

The max-pooling function with kernel size $k \times k$ outputs the maximum value among $n = k^2$ input values. To implement the max-pooling function for the homomorphically encrypted data, we should find a polynomial that approximates the max-pooling function.

We define a polynomial $M_{\alpha,n}(x_1, \cdots, x_n)$ that approximates the maximum value function $\max(x_1, \cdots, x_n)$ using $m_\alpha(a, b)$. To reduce the depth consumption of the approximate polynomial of the max-pooling function, we construct $M_{\alpha,n}$ by compositing polynomial $m_\alpha(a, b)$ with a minimal number of compositions. We use the following recursion equation to obtain the approximated maximum value among $\{x_1, \cdots, x_n\}$. We define $M_{\alpha,n}(x_1, \cdots, x_n)$ as

$$\begin{aligned}
M_{\alpha,n}(x_1, \cdots, x_n) = m_\alpha \Big(& M_{\alpha,\lfloor \frac{n}{2} \rfloor}\left(x_1, \cdots, x_{\lfloor \frac{n}{2} \rfloor}\right), \\
& M_{\alpha,n-\lfloor \frac{n}{2} \rfloor}\left(x_{\lfloor \frac{n}{2} \rfloor + 1}, \cdots, x_n\right)\Big),
\end{aligned} \tag{5}$$

where $n > 1$, and x_1 where $n = 1$.

The following Theorem shows an upper bound of approximation error of $M_{\alpha,n}(x_1, \cdots, x_n)$.

Theorem 2: For given $\alpha > 0$ and $n \in \mathbb{N}$, the polynomial $M_{\alpha,n}(x_1, \cdots, x_n)$ obtained from the recursion equation in (5) satisfies

$$|M_{\alpha,n}(x_1, \cdots, x_n) - \max(x_1, \cdots, x_n)| \le 2^{-\alpha} \lceil \log_2 n \rceil \tag{6}$$

for $x_1, \cdots, x_n \in [(\lceil \log_2 n \rceil - 1)2^{-\alpha}, 1 - (\lceil \log_2 n \rceil - 1)2^{-\alpha}]$.

Proof: The proof is given in Appendix B. ■

Approximation range. To extend the approximation range $[(\lceil \log_2 n \rceil - 1)2^{-\alpha}, 1 - (\lceil \log_2 n \rceil - 1)2^{-\alpha}]$ to $[-B, B]$ for

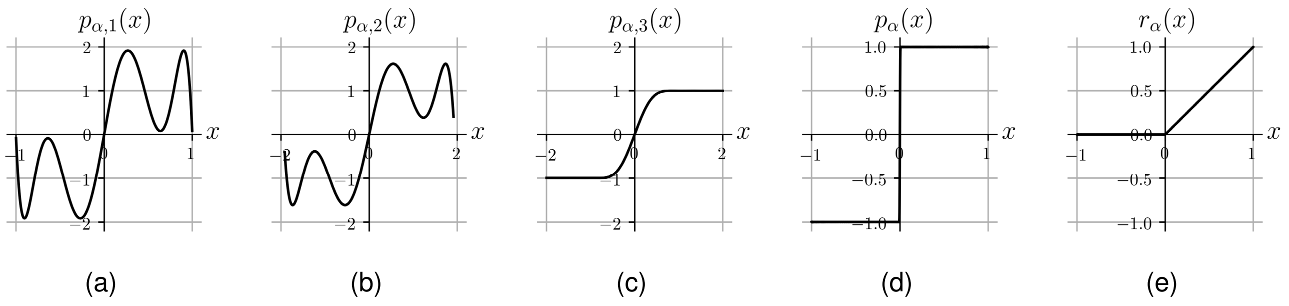


FIGURE 2. The component polynomials of $p_\alpha(x)$ and the proposed $r_\alpha(x)$ with precision parameter $\alpha = 11$. (a): the first component polynomial $p_{\alpha,1}(x)$, (b): the second component polynomial $p_{\alpha,2}(x)$, (c): the third component polynomial $p_{\alpha,3}(x)$, (d): $p_\alpha(x) = p_{\alpha,3} \circ p_{\alpha,2} \circ p_{\alpha,1}(x)$, the composition of $p_{\alpha,1}(x)$, $p_{\alpha,2}(x)$, and $p_{\alpha,3}(x)$, and (e): $r_\alpha(x) = \frac{x+xp_\alpha(x)}{2}$, which is the approximate polynomial of the ReLU function. The degrees of $p_{\alpha,1}(x)$, $p_{\alpha,2}(x)$, and $p_{\alpha,3}(x)$ are 7, 7, and 27, respectively.

$B > 1$, we propose the following polynomial $\tilde{M}_{\alpha,n,B}(x_1, \cdots, x_n)$ as an approximation polynomial:

$$\begin{aligned}\tilde{M}_{\alpha,n,B}&(x_1, \cdots, x_n)\\ &= B' \cdot \left(M_{\alpha,n}\left(\frac{x_1}{B'} + 0.5, \cdots, \frac{x_n}{B'} + 0.5\right) - 0.5\right),\end{aligned}$$

where $B' = B/(0.5 - (\lceil \log_2 n \rceil - 1)2^{-\alpha})$. Since

$$\frac{x_i}{B'} + 0.5 \in [(\lceil \log_2 n \rceil - 1)2^{-\alpha}, 1 - (\lceil \log_2 n \rceil - 1)2^{-\alpha}]$$

for $x_1, \cdots, x_n \in [-B, B]$, we have

$$\begin{aligned}&\left|\tilde{M}_{\alpha,n,B}(x_1, \cdots, x_n) - \max(x_1, \cdots, x_n)\right|\\ &= \left|B' M_{\alpha,n}\left(\frac{x_1}{B'} + 0.5, \cdots, \frac{x_n}{B'} + 0.5\right)\right.\\ &\qquad \left. -(\max(x_1, \cdots, x_n) + 0.5B')\right|\\ &= B'\left|M_{\alpha,n}\left(\frac{x_1}{B'} + 0.5, \cdots, \frac{x_n}{B'} + 0.5\right)\right.\\ &\qquad \left. - \max\left(\frac{x_1}{B'} + 0.5, \cdots, \frac{x_n}{B'} + 0.5\right)\right|\\ &\le B' \cdot 2^{-\alpha}\lceil \log_2 n \rceil\end{aligned}$$

by Theorem 2. Note that this extension is valid for $0.5 - (\lceil \log_2 n \rceil - 1)2^{-\alpha} > 0$.

C. COMPARISON BETWEEN THE MINIMAX APPROXIMATE POLYNOMIAL AND THE PROPOSED APPROXIMATE POLYNOMIAL FOR THE ReLU AND MAX-POOLING FUNCTIONS

In this subsection, we compare the proposed approximation method using $r_\alpha(x)$ and the existing minimax approximate polynomial. For simplicity, let the approximation range of the ReLU function be $[-1, 1]$ for both methods. Then, for a given degree d, the minimax approximate polynomial of degree at most d for ReLU(x) can be obtained by the Remez algorithm [34]. The precision parameter α corresponds to the logarithm of the minimax error using base two.

Fig. 3 shows the relation between precision parameter α and $\log_2 d$. By using regression, we obtain the following equation:

$$\alpha = 0.9987 \log_2 d + 2.8446. \quad (7)$$

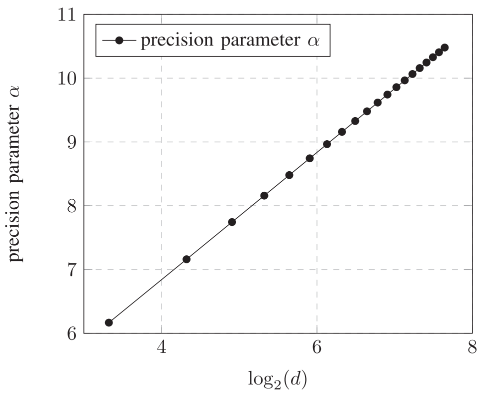


FIGURE 3. The achieved precision parameter α according to the logarithm of the polynomial degree (base two).

Because the Remez algorithm for a high-degree polynomial has the difficulties of 1) using high real number precision and 2) finding extreme points with high precision, we obtain the polynomials for only up to $d = 200$.

1) HOMOMORPHIC ReLU FUNCTION

Table 3 compares the minimax approximate polynomials and the proposed approximate polynomial $r_\alpha(x)$ for the ReLU function. The required number of non-scalar multiplications for evaluating a polynomial of degree d is $O(\sqrt{2d})$ [35], and thus, $2^{\Theta(\alpha)}$ non-scalar multiplications are required for the precision parameter α. The exact number of non-scalar multiplications is obtained using the odd baby-step giant-step algorithm for polynomials with only odd-degree terms and using the optimized baby-step giant-step algorithm for other polynomials [36].

The minimax approximate polynomials by the Remez algorithm require exponentially high degrees for high precision parameters α. These high degree polynomials lead to many non-scalar multiplications. Furthermore, numerical issues such as high precision of the real number arise when finding high-degree minimax approximate polynomials using the Remez algorithm, but research has not yet been conducted to overcome these issues. On the other hand, the proposed

TABLE 3. **Comparison of Degrees and Number of Non-scalar Multiplications for the ReLU Function. The Degrees of the minimax Approximate Polynomials and the Degrees of the Component Polynomials of $p_\alpha(x)$ are Shown. Asterisk(*) Indicates that the Degree is Obtained Using the Regression Equation of (7).**

α	Minimax approx. polynomial		Proposed approx. polynomial	
	degree	#mults	degrees	#mults
6	10	5	{3,7}	7
7	20	8	{7,7}	9
8	40	12	{7,15}	12
9	80	18	{15,15}	15
10	150	25	{7,7,13}	16
11	287*	35	{7,7,27}	19
12	575*	49	{7,15,27}	22
13	1151*	70	{15,15,27}	25
14	2304*	98	{15,27,29}	28
15	4612*	140	{29,29,29}	30

composition of polynomials can effectively reduce the number of multiplications as shown in Table 3 and does not have such numerical issues since the composition of polynomials uses low-degree component polynomials.

2) HOMOMORPHIC MAX FUNCTION

Considering $\max(a, b) = \frac{a+b+|a-b|}{2}$, we obtain the minimax approximate polynomial $p(x)$ on $[-1, 1]$ for $|x|$. Then, $m(a, b) = \frac{a+b+p(a-b)}{2}$ can be used as an approximate polynomial of the max function $\max(a, b)$. Since $\mathrm{ReLU}(x) = \frac{x+|x|}{2}$, the approximation of $|x|/2$ and $\mathrm{ReLU}(x)$ are equivalent. Thus, when the max function is approximated with the minimax approximate polynomial, an exponentially large degree is also required to achieve a high precision parameter α.

IV. THEORETICAL PERFORMANCE ANALYSIS OF APPROXIMATE DEEP LEARNING MODEL

In this section, we propose an *approximate deep learning model* for homomorphically encrypted input data, replacing the original ReLU and max-pooling functions with the proposed precise approximate polynomials $\tilde{r}_{\alpha,B}(x)$ and $\tilde{M}_{\alpha,n,B}(x_1, \cdots, x_n)$ in the original deep learning model. In addition, we analytically derive that if the pre-trained parameters of the original deep learning model are available, classification would be accurate if we adopt a proper precision parameter α. In more detail, we estimate the L-infinity norm of the difference between the inference results $\mathcal{F}(\mathbf{x})$ and $\mathcal{F}^\alpha(\mathbf{x})$, i.e., $\|\mathcal{F}^\alpha(\mathbf{x}) - \mathcal{F}(\mathbf{x})\|_\infty$. Here, $\mathcal{F}(\mathbf{x})$ and $\mathcal{F}^\alpha(\mathbf{x})$ denote the inference results of the original deep learning model $\mathcal{F}$ and the proposed approximate deep learning model $\mathcal{F}^\alpha$ with the precision parameter α, respectively. Here, $\mathcal{F}$ and $\mathcal{F}^\alpha$ share the same pre-trained parameters. The main conclusion of this section is that we can reduce the difference $\|\mathcal{F}^\alpha(\mathbf{x}) - \mathcal{F}(\mathbf{x})\|_\infty$ by increasing α, which implies that the proposed model shows almost the same performance as the original pre-trained model.

To estimate $\|\mathcal{F}^\alpha(\mathbf{x}) - \mathcal{F}(\mathbf{x})\|_\infty$, we attempt to decompose the original deep learning model, analyze each component, and finally re-combine them. First, we decompose given deep learning model $\mathcal{F}$ as $A_n \circ A_{n-1} \circ \cdots \circ A_0$, where A_i is called a *block*. Each block can be a simple function: convolution, batch normalization, activation function, and pooling function. It can also be a mixed operation that cannot be decomposed into simple operations. Here, all of the inputs and outputs of block A_i's are considered as a one-dimensional vector. Also, we denote A_i^α as the corresponding approximate polynomial operation of each block A_i with precision parameter α. If block A_i contains only a polynomial operation, A_i^α is the same as A_i since there is nothing to approximate. Then, the approximate model $\mathcal{F}^\alpha$ can be considered as a composition of approximate blocks, $A_n^\alpha \circ A_{n-1}^\alpha \circ \cdots \circ A_0^\alpha$.

If there is only a single approximate block in the approximate deep learning model, the difference in inference results can be easily determined. However, the model $\mathcal{F}^\alpha$ composed of more than two approximate blocks makes the situation more complicated. Consider an input vector $\mathbf{x}$, two consecutive blocks A_1 and A_2, and their approximation A_1^α and A_2^α. The block A_1^α makes an approximation error which can be measured as $e := \|A_1^\alpha(\mathbf{x}) - A_1(\mathbf{x})\|_\infty$. For a given input $\mathbf{x}$, the magnitude of the error e is only determined by the intrinsic feature of block A_1. However, the output error of the second block is quite different. Note that we can write the second block error as $\|A_2^\alpha(\mathbf{y} + \mathbf{e}) - A_2(\mathbf{y})\|_\infty$, where $\mathbf{y} = A_1(\mathbf{x})$ and $\mathbf{e} = A_1^\alpha(\mathbf{x}) - A_1(\mathbf{x})$. Considering $\|\mathbf{e}\|_\infty = e$, both block A_2 and the magnitude of e affect the second block error. Then, we define a new measure for each block, *error propagation function*.

In this study, all inputs $\mathbf{x}$ in the deep learning models belong to a bounded set for a given dataset. We should set the approximation range $[-B, B]$ for the proposed approximate polynomials for the ReLU and max-pooling functions. We assume that we can choose a sufficiently large B such that the inputs of all blocks fall in $[-B, B]$. That is, for a given deep learning model $\mathcal{F} = A_n \circ A_{n-1} \circ \cdots \circ A_0$ and the corresponding approximate model $\mathcal{F}^\alpha = A_n^\alpha \circ A_{n-1}^\alpha \circ \cdots \circ A_0^\alpha$, we have $\|A_i \circ \cdots \circ A_0(\mathbf{x})\|_\infty \le B$ and $\|A_i^\alpha \circ \cdots \circ A_0^\alpha(\mathbf{x})\|_\infty \le B$ for i, $0 \le i \le n-1$ and every input $\mathbf{x}$. In fact, we confirm through numerical analysis that these are satisfied for an appropriately large B in Section V. We define error propagation function as follows:

Definition 2: For a block A, an error propagation function of A, $E_A^\alpha(\cdot)$, is defined as

$$E_A^\alpha(e) := \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty \le B, \|\mathbf{e}\|_\infty \le e} \|A^\alpha(\mathbf{x} + \mathbf{e}) - A(\mathbf{x})\|_\infty,$$

where e denotes the magnitude of output error of the previous block of A.

Roughly speaking, block A^α propagates the input error e (or the output error of the previous block) to output error $E_A^\alpha(e)$. With this error propagation function, we can estimate the difference between final inference results of $\mathcal{F}$ and $\mathcal{F}^\alpha$ straightforward as in the following theorem.

Theorem 3: If the original model $\mathcal{F}$ can be decomposed as $A_n \circ A_{n-1} \circ \cdots \circ A_0$, then we have

$$\|\mathcal{F}^\alpha(\mathbf{x}) - \mathcal{F}(\mathbf{x})\|_\infty \le (E_{A_n}^\alpha \circ E_{A_{n-1}}^\alpha \circ \cdots \circ E_{A_0}^\alpha)(0)$$

for every input $\mathbf{x}$.

Proof: The proof is given in Appendix C. ■

This theorem asserts that we can achieve an upper bound of inference error of the whole model by estimating an error propagation function for each block.

We analyze the error propagation function for four types of single blocks: i) linear block, ii) ReLU block, iii) max-pooling block, and iv) softmax block, commonly used in deep neural networks for classification tasks. From now on, we call these four types of blocks *basic blocks*, and deep learning model contains only basic blocks *basic deep learning models* for convenience. Note that convolutional layers, batch normalization layers (for inference), average pooling layers, and fully connected layers are just linear combinations of input vectors and model parameters. Therefore, it is reasonable to include all these layers in a linear block. We can express linear blocks as $A(\mathbf{x}) = \mathbf{Ax} + \mathbf{b}$ for some matrix $\mathbf{A}$ and vector $\mathbf{b}$. The following lemma shows practical upper bounds of error propagation function for basic blocks. Because the kernel size of the max-pooling function is not greater than ten in most applications, we constrain the kernel size not greater than ten in the following lemma.

Lemma 1: For the given error e, the error propagation functions of four basic blocks are upper bounded as follows: (a) If A is a linear block with $A(\mathbf{x}) = \mathbf{Ax} + \mathbf{b}$, *then* $E_A^\alpha(e) \leq \|\mathbf{A}\|_\infty e$, *where* $\|\cdot\|_\infty$ *denotes the infinity norm of matrices. (b) If A is a ReLU block, then* $E_A^\alpha(e) \leq B2^{-\alpha} + e$. *(c) If A is a max-pooling block with kernel size* $k_0 \leq 10$ *and* $\alpha \geq 4$, *then* $E_A^\alpha(e) \leq 10B\lceil\log_2 k_0^2\rceil 2^{-\alpha} + e$. *(d) If A is a softmax block, then* $E_A^\alpha(e) \leq e/2$.

Proof: The proof is given in Appendix D. ■

Since the upper bounds of error propagation functions of all basic blocks have been determined, we can obtain an upper bound of the inference error theoretically as in the following theorem.

Theorem 4: For $\alpha \geq 4$ *and a basic deep learning model* $\mathcal{F}$, *there exists a constant C such that* $\|\mathcal{F}^\alpha(\mathbf{x}) - \mathcal{F}(\mathbf{x})\|_\infty \leq C2^{-\alpha}$ *for every input* $\mathbf{x}$, *where the constant C can be determined independently of the precision parameter* α.

Proof: The proof is given in Appendix E. ■

Therefore, the performance of the proposed approximate basic deep learning model can be guaranteed theoretically by Theorem 4 if the original model $\mathcal{F}$ is well trained.

Practical application of Theorem 4. Theorem 4 is valid for a basic deep learning model, and VGGNet [15] is a representative example. However, there are also various deep learning models with non-basic blocks. For example, ResNet [14] contains *residual learning building block*, which is hard to decompose into several basic blocks. Thus, the generalized version of Theorem 4 is presented in Appendix F, which is also valid for ResNet. Numerical results in Section V also support that the inference result of the original deep learning model and that of the proposed approximate deep learning model are close enough for a practical precision parameter α.

V. RESULTS OF NUMERICAL ANALYSIS

In this section, the simulation results are presented of the proposed methods for the plaintext input data[1] (Section V-A and V-B) and the ciphertext input data (Section V-C). Overall, our results show that using the high precision parameter α obtains high performance in the proposed approximate deep learning model, which confirms Theorem 4. The simulations in Section V-A and V-B are performed by NVIDIA GeForce RTX 3090 GPU along with AMD Ryzen 9 5950X 16-Core Processor CPU. In Section V-C, the simulations are performed by AMD Ryzen Threadripper PRO 3995WX at 2.096 GHz (64 cores) CPU with 512 GB RAM.

A. NUMERICAL ANALYSIS ON THE CIFAR-10 FOR THE PLAINTEXT DATA

We first use the CIFAR-10 [9] as the input data, which has 50k training images and 10k test images with 10 classes. We use ResNet and VGGNet with batch normalization as backbone models for the proposed approximate deep learning models. To verify that the proposed approximate deep learning models work well for a various number of layers, we use ResNet with 20, 32, 44, 56, and 110 layers and VGGNet with 11, 13, 16, and 19 layers proposed in [14] and [15], respectively. In order to obtain pre-trained parameters for each model, we apply the optimizer suggested in [14] on both ResNet and VGGNet for the CIFAR-10.

To adopt the proposed approximate deep learning model, the approximation range $[-B, B]$ should be determined. We examine the input values of the approximate polynomials of the ReLU and max-pooling functions and determine the value of B by adding some margin to the maximum of the input values. For the CIFAR-10, we choose $B = 50$ for both the approximate polynomials of ReLU and max-pooling functions.

The inference results of deep learning models with low-degree polynomial activation function in previous HE-friendly network and the proposed approximate ReLU are given in Table 4. First of all, neither the activation functions x^2 (suggested in [5] and [6]) nor $2^{-3}x^2 + 2^{-1}x + 2^{-2}$ (suggested in [7]) achieves good performance in our backbone models. These activation functions achieve classification accuracy around 10.0% with our pre-trained parameters. We claim that it is hard to find an appropriate low-degree polynomial for the parameters trained with exact ReLU functions, which implies the importance of the proposed precise approximate polynomial for PPML in ResNet and VGGNet. On the other hand, Table 4 shows that if we adopt a high precision parameter α, the proposed model becomes more accurate, confirming Theorem 4. We achieve the polynomial-operation-only VGG-16 model with 91.87% accuracy for the CIFAR-10 using $\alpha = 14$ without any retraining. In addi-

[1]We provide the codes of the simulations for the plaintext data (Section V-A and V-B) at: `https://github.com/snu-ccl/approxCNN`

TABLE 4. The Top-1 Accuracy of Each Approximate Deep Learning Model with Respect to α for the Plaintext CIFAR-10 Dataset. Baseline Means the Accuracy of the Original Deep Learning Model. The Accuracies of the Approximate Deep Learning Models with a Difference Less Than 5% From the Baseline Accuracy are Underlined, and Less Than 1% Are Underlined with Boldface.

Backbone	Baseline(%)	x^2 [5]	ReLU-AQ [7]	The proposed approximate ReLU polynomial $r_\alpha(x)$ $\alpha=7$	8	9	10	11	12	13	14
ResNet-20	88.36	10.00	10.00	10.00	10.00	10.00	19.96	80.91	$\underline{86.44}$	$\underline{\mathbf{88.01}}$	$\underline{\mathbf{88.41}}$
ResNet-32	89.38	10.08	10.00	10.00	10.00	10.01	36.16	$\underline{84.92}$	$\underline{88.08}$	$\underline{\mathbf{88.99}}$	$\underline{\mathbf{89.23}}$
ResNet-44	89.41	10.07	10.00	10.00	10.00	10.00	38.35	$\underline{85.75}$	$\underline{88.31}$	$\underline{\mathbf{89.31}}$	$\underline{\mathbf{89.37}}$
ResNet-56	89.72	10.23	10.00	10.00	10.00	9.94	39.48	$\underline{86.54}$	$\underline{\mathbf{89.18}}$	$\underline{\mathbf{89.53}}$	$\underline{\mathbf{89.67}}$
ResNet-110	89.87	11.11	10.00	10.00	10.00	10.04	67.02	$\underline{86.75}$	$\underline{\mathbf{89.17}}$	$\underline{\mathbf{89.62}}$	$\underline{\mathbf{89.84}}$
VGG-11	90.17	10.22	10.00	10.77	11.74	15.61	29.39	66.60	85.22	$\underline{88.95}$	$\underline{\mathbf{89.91}}$
VGG-13	91.90	10.01	10.00	11.32	11.57	14.37	36.84	77.55	$\underline{88.72}$	$\underline{90.66}$	$\underline{\mathbf{91.37}}$
VGG-16	91.99	10.01	10.00	10.02	10.06	10.76	32.93	78.27	$\underline{89.69}$	$\underline{\mathbf{91.13}}$	$\underline{\mathbf{91.87}}$
VGG-19	91.75	9.99	10.00	10.18	13.84	18.18	43.56	80.40	$\underline{89.42}$	$\underline{\mathbf{91.10}}$	$\underline{\mathbf{91.61}}$

TABLE 5. The Top-1 Accuracy of Each Approximate Deep Learning Model for the Plaintext ImageNet Dataset with $\alpha = 14$. The Accuracies Are Underlined and Boldfaced in the Same Manner as in Table 4.

Backbone	B ReLU	Max-pooling	Baseline (%)	Proposed (%)
ResNet-152 [14]	100.0	10.0	78.31	$\underline{77.52}$
VGG-19 [15]	100.0	50.0	74.22	$\underline{\mathbf{73.56}}$
GoogLeNet [16]	50.0	50.0	68.13	$\underline{69.81}$
Inception-v3 [37]	200.0	50.0	69.54	$\underline{\mathbf{69.48}}$

tion, the proposed approximate deep learning model performs almost similarly (1% difference) to the original deep learning model when $\alpha = 12$–14.

As we mentioned, obtaining high-degree polynomial with the Remez algorithm with high precision has some difficulties. Therefore, we did not compare between the performance of the approximate deep learning model using single minimax polynomial introduced in Section III-C and the proposed approximate polynomial.

B. NUMERICAL ANALYSIS ON THE ImageNet FOR THE PLAINTEXT DATA

We also analyze the classification accuracy of the proposed approximate deep learning model for the ImageNet dataset [8], which has 1.28M training images and 50k validation images with 1000 classes. We simulate various models whose pre-trained parameters are available for inference of the ImageNet. The models that we use and their inference results are summarized in Table 5. The values of B we choose for each model are also denoted in Table 5. Without any training, we easily achieve a polynomial-operation-only model with an accuracy of 77.52% for the ImageNet. From this result, it can be seen that even if the dataset is large, the proposed model shows high performance, similar to the original model.

C. NUMERICAL ANALYSIS ON THE CIFAR-10 FOR THE ENCRYPTED DATA

To justify the simulation results of classification accuracy for the plaintext data, we encrypt the CIFAR-10 dataset with the RNS-CKKS scheme and perform ResNet models using the SEAL library [38] as described in [31]. Since the operations performed for encrypted data are all homomorphic operations, the security of performing ResNet models with the encrypted images can guarantee the security of RNS-CKKS scheme.

TABLE 6. The Top-1 Accuracy and a Classification Runtime for a Single Image of Each Approximate Deep Learning Model for the Ciphertext CIFAR-10 Dataset with $\alpha = 13$. The Accuracies Are Underlined and Boldfaced in the Same Manner as in Table 4.

Backbone	Baseline(%)	Proposed(%) (plaintext)	Proposed(%) (ciphertext)	Time(s)
ResNet-20	88.36	$\underline{\mathbf{88.01}}$	$\underline{\mathbf{87.90}}$	2,892
ResNet-32	89.38	$\underline{\mathbf{88.99}}$	$\underline{88.20}$	4,764

To perform the RNS-CKKS scheme, we set proper security parameters, such as the degree of ciphertexts N or modulus Q, to achieve enough security level. We follow the same parameter setting of the experiment in [31]. We set $N = 2^{16}$ and Q as a product of 51-bit primes. Also, we encrypt a message with the Hamming weight of the secret key 128. With these settings, we achieve the standard 128-bit security level.

We choose the precision parameter $\alpha = 13$ and the approximation range $[-50, 50]$. We classify the whole 10k test images for the ResNet-20 model, and the first 1,000 test images for the ResNet-32. Table 6 summarizes the results. For the ResNet-20 model, we obtain top-1 accuracy of 87.90%, which has a difference of less than 1% from the baseline accuracy of 88.36%. In addition, the top-1 accuracy of 87.90% is very close to the accuracy of 88.01% obtained from simulation for plaintext data, which implies that the simulation for the plaintext data is also valid for the encrypted data. For the ResNet-32 model, we obtain top-1 accuracy of 88.20%, which has a difference of less than 2% from the baseline accuracy of 89.38%. The simulation results support that the proposed approximate deep learning models perform well on the encrypted image dataset and achieve the comparable classification accuracy of the original models.

D. DISCUSSION

Although the numerical analyses in Sections V-A and V-B were performed for plaintext data rather than encrypted data, these simulations can all be performed on the RNS-CKKS since all non-arithmetic operations have been replaced with polynomial operations. In the simulation for the encrypted data, bootstrapping error should also be considered as a potential factor that may reduce the classification accuracy on the RNS-CKKS scheme. However, a recent developed high-precision bootstrapping technique [10] can achieve up to 40-bit precision. Thus, the accuracy obtained from simulation on the encrypted data using sufficiently high-precision bootstrapping will be very close to that obtained in the simulation for plaintext data. The implementation results of ResNet on the encrypted data in Section V-C supports the validity of simulation for the plaintext data.

VI. CONCLUSION AND FUTURE WORKS

We proposed the polynomials that precisely approximate the ReLU and max-pooling functions using a composition of minimax approximate polynomials of small degrees for PPML using FHE. We showed theoretically and numerically that if the precision parameter α is large enough, the difference between the inference result of the original and the proposed approximate models becomes small enough. Furthermore, for the first time for PPML using word-wise HE, we achieved high performance for ImageNet classification using the proposed approximate polynomials. Finally, we performed the ResNet models with encrypted CIFAR-10 dataset to show the validity of simulation for the plaintext data.

For utilizing PPML using FHE in the actual industry, we are considering two issues as our future work. One of the important topics is to optimize the components of the deep learning model so that inference can be effectively performed. Since the runtime for PPML inference is still impractical (as seen in Table 6,) the proposed approximate deep learning model needs to be optimized to reduce the runtime. Acceleration using hardware accelerators, optimization of the RNS-CKKS algorithm, and more efficient approximation methods for polynomials should be considered. Also, we will apply these optimizations and perform the classification with encrypted data not only for the ResNet but also for other popular networks such as VGGNet. Another topic of our future work is the training of deep learning models using encrypted data. In the real world of industries utilizing cloud computing, there may not be much accessible data for training models due to privacy issues. There are many interesting points to be considered for training deep learning models using encrypted data (e.g., stochastic gradient descent or backpropagation), and these points will be significant future research topics.

APPENDIX A
PROOF OF THEOREM 1

From the definition, $p_\alpha(x)$ satisfies the following inequality:

$$|m_\alpha(a,b) - \max(a,b)| = |\frac{(a+b)+(a-b)p_\alpha(a-b)}{2} - \max(a,b)| \le 2^{-\alpha},$$

for $a, b \in [0,1]$. Then, for $x \in [0,1]$, we have

$$|r_\alpha(x) - \mathrm{ReLU}(x)| = |\frac{x + xp_\alpha(x)}{2} - x| = |m_\alpha(x,0) - \max(x,0)| \le 2^{-\alpha}.$$

In addition, for $x \in [-1,0]$, we have

$$|r_\alpha(x) - \mathrm{ReLU}(x)| = |\frac{x + xp_\alpha(x)}{2}| = |\frac{-x + xp_\alpha(x)}{2} + x| = |m_\alpha(0,-x) - \max(0,-x)| \le 2^{-\alpha}.$$

Thus, we have $|r_\alpha(x) - \mathrm{ReLU}(x)| \le 2^{-\alpha}$ for $x \in [-1,1]$.

APPENDIX B
PROOF OF THEOREM 2

To prove this theorem, we require two lemmas.

Lemma 2: For $n \in \mathbb{N}$, let S and T satisfy

$$(\lceil \log_2 n \rceil - 1)2^{-\alpha} \le S < T \le 1 - (\lceil \log_2 n \rceil - 1)2^{-\alpha}.$$

Then, for $x_1, x_2, \cdots, x_n \in [S,T]$, the following is satisfied:

$$M_{\alpha,n}(x_1, \cdots, x_n) \in [S - \lceil \log_2 n \rceil 2^{-\alpha}, T + \lceil \log_2 n \rceil 2^{-\alpha}].$$

Proof: We will use mathematical induction to show that Lemma 2 holds for all $n \in N$. For $n = 1$, it is trivial because $M_{\alpha,n}(x_1, \cdots, x_n) \in [S - \lceil \log_2 n \rceil 2^{-\alpha}, T + \lceil \log_2 n \rceil 2^{-\alpha}]$ if and only if $x_1 \in [S,T]$.

For $n = 2$, we have $0 \le S < T \le 1$. We have to show that $M_{\alpha,2}(x_1,x_2) = m_\alpha(x_1,x_2) \in [S - 2^{-\alpha}, T + 2^{-\alpha}]$ for $x_1, x_2 \in [S,T]$. We note that

$$|m_\alpha(a,b) - \max(a,b)| \le 2^{-\alpha} \quad \text{for } a, b \in [0,1]. \tag{8}$$

Because $|m_\alpha(x_1,x_2) - \max(x_1,x_2)| \le 2^{-\alpha}$, we have

$$\begin{aligned} -2^{-\alpha} + S &\le m_\alpha(x_1,x_2) - \max(x_1,x_2) + S \\ &\le m_\alpha(x_1,x_2) - \max(x_1,x_2) + \max(x_1,x_2) \\ &= m_\alpha(x_1,x_2). \end{aligned}$$

Also, we have

$$\begin{aligned} m_\alpha(x_1,x_2) &= m_\alpha(x_1,x_2) - \max(x_1,x_2) + \max(x_1,x_2) \\ &\le 2^{-\alpha} + \max(x_1,x_2) \\ &\le 2^{-\alpha} + T. \end{aligned}$$

Thus, Lemma 2 holds for $n = 2$. Now, we assume that Lemma 2 holds for n, $1 \le n \le m-1$ for some $m \ge 3$. It is enough to show that Lemma 2 also holds for $n = m$.

(i) $m = 2k$

We have

$$(\lceil \log_2 2k \rceil - 1)2^{-\alpha} \le S < T \le 1 - (\lceil \log_2 2k \rceil - 1)2^{-\alpha}, \tag{9}$$

which is equivalent to $(\lceil \log_2 k \rceil)2^{-\alpha} \le S < T \le 1 - (\lceil \log_2 k \rceil)2^{-\alpha}$. Then, we have to show that

$$M_{\alpha,2k}(x_1, \cdots, x_{2k}) \in [S - \lceil \log_2 2k \rceil 2^{-\alpha}, T + \lceil \log_2 2k \rceil 2^{-\alpha}],$$

for $x_1, \cdots, x_{2k} \in [S,T]$. Because Lemma 2 holds for $n = k$ by the intermediate induction assumption, we have

$$\begin{aligned} &M_{\alpha,k}(x_1, \cdots, x_k), M_{\alpha,k}(x_{k+1}, \cdots, x_{2k}) \\ &\quad \in [S - \lceil \log_2 k \rceil 2^{-\alpha}, T + \lceil \log_2 k \rceil 2^{-\alpha}] \end{aligned}$$

From the inequality in (9), we have $0 \le S - \lceil \log_2 k \rceil 2^{-\alpha}$ and $T + \lceil \log_2 k \rceil 2^{-\alpha} \le 1$. Thus, we have

$$M_{\alpha,k}(x_1, \cdots, x_k), M_{\alpha,k}(x_{k+1}, \cdots, x_{2k}) \subseteq [0,1].$$

Then, from Lemma 2 for $n = 2$, we have

$$\begin{aligned} &M_{\alpha,2k}(x_1, \cdots, x_{2k}) \\ &= m_\alpha(M_{\alpha,k}(x_1, \cdots, x_k), M_{\alpha,k}(x_{k+1}, \cdots, x_{2k})) \\ &\in [S - \lceil \log_2 k \rceil 2^{-\alpha} - 2^{-\alpha}, T + \lceil \log_2 k \rceil 2^{-\alpha} + 2^{-\alpha}] \\ &= [S - \lceil \log_2 2k \rceil 2^{-\alpha}, T + \lceil \log_2 2k \rceil 2^{-\alpha}]. \end{aligned}$$

Thus, Lemma 2 holds for $n = m = 2k$.
(ii) $m = 2k+1$
We have

$$(\lceil \log_2(2k+1)\rceil - 1)2^{-\alpha} \le S < T \le 1 - (\lceil \log_2(2k+1)\rceil - 1)2^{-\alpha}.$$

This is equivalent to

$$(\lceil \log_2(k+1)\rceil)2^{-\alpha} \le S < T \le 1 - (\lceil \log_2(k+1)\rceil)2^{-\alpha} \tag{10}$$

because $\lceil \log_2(2k+1)\rceil = \lceil \log_2(2k+2)\rceil$ for every integer $k \ge 1$. Then, we have to show that

$$M_{\alpha,2k+1}(x_1,\cdots,x_{2k+1}) \in [S - \lceil \log_2(2k+1)\rceil 2^{-\alpha}, T + \lceil \log_2(2k+1)\rceil 2^{-\alpha}],$$

for $x_1,\cdots,x_{2k+1} \in [S,T]$. Because Lemma 2 holds for $n = k$ and $n = k+1$ by the intermediate induction assumption, we have

$$\begin{aligned}&M_{\alpha,k}(x_1,\cdots,x_k)\\ &\in [S - \lceil \log_2 k\rceil 2^{-\alpha}, T + \lceil \log_2 k\rceil 2^{-\alpha}] \text{ and}\\ &M_{\alpha,k+1}(x_{k+1},\cdots,x_{2k+1})\\ &\in [S - \lceil \log_2(k+1)\rceil 2^{-\alpha}, T + \lceil \log_2(k+1)\rceil 2^{-\alpha}].\end{aligned}$$

From the inequality in (10), we have $0 \le S - \lceil \log_2(k+1)\rceil 2^{-\alpha}$ and $T + \lceil \log_2(k+1)\rceil 2^{-\alpha} \le 1$. Thus, we have

$$\begin{aligned}&M_{\alpha,k}(x_1,\cdots,x_k), M_{\alpha,k+1}(x_{k+1},\cdots,x_{2k+1})\\ &\in [S - \lceil \log_2(k+1)\rceil 2^{-\alpha}, T + \lceil \log_2(k+1)\rceil 2^{-\alpha}]\\ &\subseteq [0,1].\end{aligned}$$

Then, from Lemma 2 for $n = 2$, we have

$$\begin{aligned}&M_{\alpha,2k+1}(x_1,\cdots,x_{2k+1})\\ &= m_\alpha(M_{\alpha,k}(x_1,\cdots,x_k), M_{\alpha,k+1}(x_{k+1},\cdots,x_{2k+1}))\\ &\in [S - \lceil \log_2(k+1)\rceil 2^{-\alpha} - 2^{-\alpha},\\ &\qquad T + \lceil \log_2(k+1)\rceil 2^{-\alpha} + 2^{-\alpha}]\\ &= [S - \lceil \log_2(2k+2)\rceil 2^{-\alpha}, T + \lceil \log_2(2k+2)\rceil 2^{-\alpha}]\\ &= [S - \lceil \log_2(2k+1)\rceil 2^{-\alpha}, T + \lceil \log_2(2k+1)\rceil 2^{-\alpha}].\end{aligned}$$

Thus, Lemma 2 holds for $n = m = 2k+1$.

Then, Lemma 2 holds for all $n \ge 1$ by mathematical induction. ■

Lemma 3: For $a, b, c, d \in \mathbb{R}$, we have

$$|\max(a,b) - \max(c,d)| \le \max(|a-c|, |b-d|) \tag{11}$$

Proof: Let $\max(a,b) = a$ without loss of generality. We denote the left-hand side and right-hand side of the inequality in (11) by LHS and RHS, respectively.

1) $\max(c,d) = c$
We have LHS $= |a-c|$. Thus, the lemma holds in this case.
2) $\max(c,d) = d$
We have LHS $= |a-d|$.
   a) $a \ge d$
   We have $a \ge d \ge c$. Thus, we have LHS $= |a-d| \le |a-c| \le$ RHS.
   b) $a < d$
   We have $b \le a < d$. Thus, we have LHS $= |a-d| \le |b-d| \le$ RHS.

Thus, the lemma is proved. ■

Now, we prove the theorem using Lemma 2 and 3. For convenience, we write the interval $[(\lceil \log_2 n\rceil - 1)2^{-\alpha}, 1 - (\lceil \log_2 n\rceil - 1)2^{-\alpha}]$ as I_n. We have to show the inequality (6) for $x_1,\cdots,x_n \in I_n$.

We use mathematical induction to show that the inequality in (6) holds for all $n \in \mathbb{N}$. First, for $n = 1$, we have

$$|M_{\alpha,1}(x_1) - \max(x_1)| = |x_1 - x_1| = 0 = 2^{-\alpha}\lceil \log_2 1\rceil.$$

Therefore, the inequality in (6) holds for $n = 1$. We assume that the inequality holds for all n, $1 \le n \le m-1$. Then, it is enough to show that the inequality in (6) also holds for $n = m$. Suppose that $x_1,\cdots,x_m \in I_m$.

(i) $m = 2k$
We have to show that

$$|M_{\alpha,2k}(x_1,\cdots,x_{2k}) - \max(x_1,\cdots,x_{2k})| \le 2^{-\alpha}\lceil \log_2 2k\rceil$$

for $x_1,\cdots,x_{2k} \in I_{2k}$. We have $M_{\alpha,2k}(x_1,\cdots,x_{2k}) = m_\alpha(M_{\alpha,k}(x_1,\cdots,x_k), M_{\alpha,k}(x_{k+1},\cdots,x_{2k}))$ from the recursion formula (5).

Let $P = \max(x_1,\cdots,x_k)$, $Q = \max(x_{k+1},\cdots,x_{2k})$, $\tilde{P} = M_{\alpha,k}(x_1,\cdots,x_k)$, and $\tilde{Q} = M_{\alpha,k}(x_{k+1},\cdots,x_{2k})$. Since $x_1,\cdots,x_{2k} \in I_{2k} \subseteq I_k$, we can apply the intermediate induction assumption for $n = k$ as

$$\begin{aligned}|\tilde{P} - P| &= |M_{\alpha,k}(x_1,\cdots,x_k) - \max(x_1,\cdots,x_k)|\\ &\le 2^{-\alpha}\lceil \log_2 k\rceil,\\ |\tilde{Q} - Q| &= |M_{\alpha,k}(x_{k+1},\cdots,x_{2k}) - \max(x_{k+1},\cdots,x_{2k})|\\ &\le 2^{-\alpha}\lceil \log_2 k\rceil.\end{aligned}$$

The left-hand side of the inequality in (6) for $n = m$ becomes $|m_\alpha(\tilde{P},\tilde{Q}) - \max(P,Q)|$. From Lemma 2 for $n = k$, we have $\tilde{P},\tilde{Q} \in [0,1]$. Then, we have

$$\begin{aligned}&|m_\alpha(\tilde{P},\tilde{Q}) - \max(P,Q)|\\ &\le |m_\alpha(\tilde{P},\tilde{Q}) - \max(\tilde{P},\tilde{Q})| + |\max(\tilde{P},\tilde{Q}) - \max(P,Q)|\\ &\le |m_\alpha(\tilde{P},\tilde{Q}) - \max(\tilde{P},\tilde{Q})| + \max(|\tilde{P} - P|, |\tilde{Q} - Q|)\\ &\qquad \text{(from Lemma 3)}\\ &\le 2^{-\alpha} + \max(|\tilde{P} - P|, |\tilde{Q} - Q|) \quad \text{(from (8))}\\ &\le 2^{-\alpha} + 2^{-\alpha}\lceil \log_2 k\rceil = 2^{-\alpha}\lceil \log_2 2k\rceil = 2^{-\alpha}\lceil \log_2 m\rceil.\end{aligned}$$

(ii) $m = 2k+1$
We have to show that

$$|M_{\alpha,2k+1}(x_1,\cdots,x_{2k+1}) - \max(x_1,\cdots,x_{2k+1})| \le 2^{-\alpha}\lceil \log_2(2k+1)\rceil$$

for $x_1,\cdots,x_{2k+1} \in I_{2k+1}$. From the recursion formula (5), we have $M_{\alpha,2k+1}(x_1,\cdots,x_{2k+1}) = m_\alpha(M_{\alpha,k}(x_1,\cdots,x_k), M_{\alpha,k+1}(x_{k+1},\cdots,x_{2k+1}))$.

Let $P = \max(x_1, \cdots, x_k)$, $Q = \max(x_{k+1}, \cdots, x_{2k+1})$, $\tilde{P} = M_{\alpha,k}(x_1, \cdots, x_k)$, and $\tilde{Q} = M_{\alpha,k+1}(x_{k+1}, \cdots, x_{2k+1})$. Since $x_1, \cdots, x_k \in I_{2k+1} \subseteq I_k$ and $x_{k+1}, \cdots, x_{2k+1} \in I_{2k+1} \subseteq I_{k+1}$, we can apply the induction assumption for $n = k$ and $n = k+1$ as

$$\begin{aligned}|\tilde{P} - P| &= |M_{\alpha,k}(x_1, \cdots, x_k) - \max(x_1, \cdots, x_k)| \\ &\leq 2^{-\alpha}\lceil \log_2 k\rceil, \\ |\tilde{Q} - Q| &= |M_{\alpha,k+1}(x_{k+1}, \cdots, x_{2k+1}) \\ &\qquad - \max(x_{k+1}, \cdots, x_{2k+1})| \\ &\leq 2^{-\alpha}\lceil \log_2(k+1)\rceil.\end{aligned}$$

The left-hand side of the inequality in (6) for $n = m$ becomes $|m_\alpha(\tilde{P}, \tilde{Q}) - \max(P, Q)|$. From Lemma 2 for $n = k$ and $n = k+1$, we have $\tilde{P}, \tilde{Q} \in [0, 1]$. Then, we have

$$\begin{aligned}&|m_\alpha(\tilde{P}, \tilde{Q}) - \max(P, Q)| \\ &\leq |m_\alpha(\tilde{P}, \tilde{Q}) - \max(\tilde{P}, \tilde{Q})| + |\max(\tilde{P}, \tilde{Q}) - \max(P, Q)| \\ &\leq |m_\alpha(\tilde{P}, \tilde{Q}) - \max(\tilde{P}, \tilde{Q})| + \max(|\tilde{P} - P|, |\tilde{Q} - Q|) \\ &\qquad \text{(from Lemma 3)} \\ &\leq 2^{-\alpha} + \max(|\tilde{P} - P|, |\tilde{Q} - Q|) \quad \text{(from (8))} \\ &\leq 2^{-\alpha} + 2^{-\alpha}\lceil \log_2(k+1)\rceil \\ &= 2^{-\alpha}\lceil \log_2(2k+2)\rceil = 2^{-\alpha}\lceil \log_2(2k+1)\rceil = 2^{-\alpha}\lceil \log_2 m\rceil\end{aligned}$$

since $\lceil \log_2(2k+2)\rceil = \lceil \log_2(2k+1)\rceil$ for every integer $k \geq 1$. Thus, the inequality in (6) holds for $n = m$, and the theorem is proved by mathematical induction.

APPENDIX C
PROOF OF THEOREM 3

For convenience, we write the composition of the functions $\{f_i\}_{i=0}^{j}, f_j \circ \cdots \circ f_0$ as $\mathsf{C}_{i=0}^{j} f_i$ in this proof. Equivalently, we have to show that

$$\left\| \left[\mathop{\mathsf{C}}_{i=0}^{n} A_i^\alpha\right](\mathbf{x}) - \left[\mathop{\mathsf{C}}_{i=0}^{n} A_i\right](\mathbf{x}) \right\|_\infty \leq \left[\mathop{\mathsf{C}}_{i=0}^{n} E_{A_i}^\alpha\right](0). \tag{12}$$

We will prove it by mathematical induction. First, we will prove for $n = 0$.

$$\begin{aligned}\|A_0^\alpha(\mathbf{x}) - A_0(\mathbf{x})\|_\infty &\leq \sup_{\|\mathbf{x}\|_\infty \leq B} \|A_0^\alpha(\mathbf{x}) - A_0(\mathbf{x})\|_\infty \\ &= \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty \leq B, \|\mathbf{e}\|_\infty \leq 0} \|A_0^\alpha(\mathbf{x}+\mathbf{e}) - A_0(\mathbf{x})\|_\infty = E_{A_0}^\alpha(0).\end{aligned}$$

Thus, the inequality in (12) holds for $n = 0$.

Next, we assume that the inequality in (12) holds for $n = k$, that is,

$$\left\| \left[\mathop{\mathsf{C}}_{i=0}^{k} A_i^\alpha\right](\mathbf{x}) - \left[\mathop{\mathsf{C}}_{i=0}^{k} A_i\right](\mathbf{x}) \right\|_\infty \leq \left[\mathop{\mathsf{C}}_{i=0}^{k} E_{A_i}^\alpha\right](0).$$

for some $k \geq 0$. It is enough to show that the inequality in (12) also holds for $n = k+1$, that is,

$$\left\| \left[\mathop{\mathsf{C}}_{i=0}^{k+1} A_i^\alpha\right](\mathbf{x}) - \left[\mathop{\mathsf{C}}_{i=0}^{k+1} A_i\right](\mathbf{x}) \right\|_\infty \leq \left[\mathop{\mathsf{C}}_{i=0}^{k+1} E_{A_i}^\alpha\right](0).$$

We have

$$\begin{aligned}&\left\| \left[\mathop{\mathsf{C}}_{i=0}^{k+1} A_i^\alpha\right](\mathbf{x}) - \left[\mathop{\mathsf{C}}_{i=0}^{k+1} A_i\right](\mathbf{x}) \right\|_\infty \\ &= \left\| A_{k+1}^\alpha\left(\left[\mathop{\mathsf{C}}_{i=0}^{k} A_i\right](\mathbf{x}) + (\left[\mathop{\mathsf{C}}_{i=0}^{k} A_i^\alpha\right](\mathbf{x}) - \left[\mathop{\mathsf{C}}_{i=0}^{k} A_i\right](\mathbf{x}))\right) \right. \\ &\qquad \left. - \left[\mathop{\mathsf{C}}_{i=0}^{k+1} A_i\right](\mathbf{x}) \right\|_\infty.\end{aligned}$$

Let $\mathbf{x}' = \left[\mathop{\mathsf{C}}_{i=0}^{k} A_i\right](\mathbf{x})$ and $\mathbf{e}' = \left[\mathop{\mathsf{C}}_{i=0}^{k} A_i^\alpha\right](\mathbf{x}) - \left[\mathop{\mathsf{C}}_{i=0}^{k} A_i\right](\mathbf{x})$. Because $\|\mathbf{x}' + \mathbf{e}'\|_\infty = \left\| \left[\mathop{\mathsf{C}}_{i=0}^{k} A_i^\alpha\right](\mathbf{x}) \right\|_\infty \leq B$, we have

$$\begin{aligned}&\left\| \left[\mathop{\mathsf{C}}_{i=0}^{k+1} A_i^\alpha\right](\mathbf{x}) - \left[\mathop{\mathsf{C}}_{i=0}^{k+1} A_i\right](\mathbf{x}) \right\|_\infty \\ &= \|A_{k+1}^\alpha(\mathbf{x}' + \mathbf{e}') - A_{k+1}(\mathbf{x}')\|_\infty \\ &\leq \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty \leq B, \|\mathbf{e}\|_\infty \leq \|\mathbf{e}'\|_\infty} \|A_{k+1}^\alpha(\mathbf{x}+\mathbf{e}) - A_{k+1}(\mathbf{x})\|_\infty \\ &= E_{A_{k+1}}^\alpha(\|\mathbf{e}'\|_\infty) \\ &= E_{A_{k+1}}^\alpha\left(\left\| \left[\mathop{\mathsf{C}}_{i=0}^{k} A_i^\alpha\right](\mathbf{x}) - \left[\mathop{\mathsf{C}}_{i=0}^{k} A_i\right](\mathbf{x}) \right\|_\infty \right) \\ &\leq E_{A_{k+1}}^\alpha\left(\left[\mathop{\mathsf{C}}_{i=0}^{k} E_{A_i}^\alpha\right](0) \right) \quad (\because E_{A_{k+1}}^\alpha(\cdot): \text{ increasing}) \\ &= \left[\mathop{\mathsf{C}}_{i=0}^{k+1} E_{A_i}^\alpha\right](0).\end{aligned}$$

Thus, the inequality in (12) holds for $n = k+1$, and the theorem is proved by mathematical induction.

APPENDIX D
PROOF of LEMMA 1

(a) For $A(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, $A^\alpha = A$. Then $A^\alpha(\mathbf{x}+\mathbf{e}) - A(\mathbf{x}) = \mathbf{A}\mathbf{e}$. Therefore, $E_A^\alpha(e) = \sup_{\|\mathbf{e}\|_\infty \leq e} \|\mathbf{A}\mathbf{e}\|_\infty \leq \sup_{\|\mathbf{e}\|_\infty \leq e} \|\mathbf{A}\|_\infty \|\mathbf{e}\|_\infty = \|\mathbf{A}\|_\infty e$.

(b) For $A(\mathbf{x}) = \mathrm{ReLU}(\mathbf{x})$, $A^\alpha(\mathbf{x}) = \tilde{r}_{\alpha,B}(\mathbf{x})$. Therefore,

$$\begin{aligned}E_A^\alpha(e) &= \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty \leq B, \|\mathbf{e}\|_\infty \leq e} \|\tilde{r}_{\alpha,B}(\mathbf{x}+\mathbf{e}) - \mathrm{ReLU}(\mathbf{x})\|_\infty \\ &\leq \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty \leq B, \|\mathbf{e}\|_\infty \leq e} (\|\tilde{r}_{\alpha,B}(\mathbf{x}+\mathbf{e}) - \mathrm{ReLU}(\mathbf{x}+\mathbf{e})\|_\infty \\ &\qquad + \|\mathrm{ReLU}(\mathbf{x}+\mathbf{e}) - \mathrm{ReLU}(\mathbf{x})\|_\infty) \\ &\leq \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty \leq B, \|\mathbf{e}\|_\infty \leq e} (B \cdot 2^{-\alpha} + \|\mathbf{e}\|_\infty) = B \cdot 2^{-\alpha} + e.\end{aligned}$$

(c) If A is a max-pooling block with kernel size k_0, $\sup_{\|\mathbf{x}+\mathbf{e}\|_\infty \leq B, \|\mathbf{e}\|_\infty \leq e} \|A^\alpha(\mathbf{x}+\mathbf{e}) - A(\mathbf{x})\|_\infty$ becomes

$$\begin{aligned}\sup_{\|\mathbf{x}+\mathbf{e}\|_\infty \leq B, \|\mathbf{e}\|_\infty \leq e} &|\tilde{M}_{\alpha,k_0^2,B}(x_1 + e_1, \cdots, x_{k_0^2} + e_{k_0^2}) \\ &\qquad - \max(x_1, \cdots, x_{k_0^2})|,\end{aligned}$$

where $\mathbf{x} = (x_1, \cdots, x_{k_0^2})$ and $\mathbf{e} = (e_1, \cdots, e_{k_0^2})$. Similar technique of proof for (b) and from Theorem 2, we have $E_A^\alpha(e) \leq B'\lceil \log_2 k_0^2\rceil 2^{-\alpha} + e$, where $B' = B/(0.5 - (\lceil \log_2 k_0^2\rceil - 1)2^{-\alpha})$. Since $k_0 \leq 10$ and $\alpha \geq 4$, $(\lceil \log_2 k_0^2\rceil - 1)2^{-\alpha} < 0.4$, therefore $B' < 10B$, which leads the conclusion.

(d) Let us denote softmax block A as an $\mathbb{R}^N \to \mathbb{R}^N$ function with $A(\mathbf{x}) = (\frac{\exp(x_i)}{\sum_j \exp(x_j)})_{1\le i\le N}$ for $\mathbf{x} = (x_1, \cdots, x_N)$. Then mean value theorem for multivariate variable functions gives

$$\|A(\mathbf{x}') - A(\mathbf{x})\|_\infty \le \sup_{\mathbf{z}\in[\mathbf{x}',\mathbf{x}]} \|\mathbf{J}(\mathbf{z})\|_\infty \cdot \|\mathbf{x}' - \mathbf{x}\|_\infty,$$

where $\mathbf{J}(\mathbf{z})$ denotes the Jacobian matrix of softmax block A. For a given vector $\mathbf{z} = (z_1, \cdots, z_N) \in \mathbb{R}^N$, the infinity norm of $\mathbf{J}(\mathbf{z})$ is given as $\|\mathbf{J}(\mathbf{z})\|_\infty = \max_i \sum_k |\frac{\partial}{\partial z_k}\frac{\exp(z_i)}{\sum_j \exp(z_j)}|$. Note that

$$\left|\frac{\partial}{\partial z_k}\frac{\exp(z_i)}{\sum_j \exp(z_j)}\right| = \begin{cases} \dfrac{\exp(z_i)\exp(z_k)}{(\sum_j \exp(z_j))^2}, & k \ne i \\ \dfrac{\exp(z_i)\sum_{j\ne i}\exp(z_j)}{(\sum_j \exp(z_j))^2}, & k = i \end{cases}$$

and thus

$$\begin{aligned}\sum_k \left|\frac{\partial}{\partial z_k}\frac{\exp(z_i)}{\sum_j \exp(z_j)}\right| &= \frac{\exp(z_i)\sum_{k\ne i}\exp(z_k)}{(\sum_j \exp(z_j))^2} + \frac{\exp(z_i)\sum_{j\ne i}\exp(z_j)}{(\sum_j \exp(z_j))^2} \\ &= 2p_i(1-p_i),\end{aligned}$$

where $p_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$. Therefore, $\|\mathbf{J}(\mathbf{z})\|_\infty = \max_i 2p_i(1-p_i) \le 1/2$ for any vector $\mathbf{z} \in \mathbb{R}^N$, and this gives

$$\|A^\alpha(\mathbf{x}+\mathbf{e}) - A(\mathbf{x})\|_\infty = \|A(\mathbf{x}+\mathbf{e}) - A(\mathbf{x})\|_\infty \le \|\mathbf{e}\|_\infty/2,$$

which leads the conclusion.

APPENDIX E
PROOF OF THEOREM 4

From Theorem 3, it is enough to show that

$$(E^\alpha_{A_n} \circ \cdots \circ E^\alpha_{A_0})(0) \le C2^{-\alpha} \tag{13}$$

for some constant C and $n \ge 0$. To show that this inequality holds for all $n \ge 0$, we use mathematical induction. First, for $n = 0$, assume that $\mathcal{F}$ has one block A. The upper bounds of $E^\alpha_A(0)$ for the four basic blocks suggested in Lemma 1 have forms of $C_0 \cdot 2^{-\alpha}$, where C_0 can be zero. Thus, the inequality in (13) holds for some constant C and $n = 0$.

Then, we assume that the inequality in (13) holds for $n = k$ and some constant C_k, that is, $(E^\alpha_{A_k} \circ\cdots\circ E^\alpha_{A_0})(0) \le C_k 2^{-\alpha}$ for some C_k. Then, it is enough to show that the inequality in (13) holds for $n = k+1$ and some constant C_{k+1}. $E^\alpha_{A_{k+1}}(C_k 2^\alpha)$ is not greater than

(i) $\|\mathbf{A}\|_\infty C_k 2^{-\alpha}$ when A_{k+1} is a linear block with $A(\mathbf{x}) = \mathbf{Ax} + \mathbf{b}$,
(ii) $(B + C_k)2^{-\alpha}$ when A_{k+1} is a ReLU block,
(iii) $(10B\lceil\log_2 k_0^2\rceil + C_k)2^{-\alpha}$ when A_{k+1} is a max-pooling block with kernel size k_0,
(iv) $\frac{1}{2}C_k 2^{-\alpha}$ when A_{k+1} is a softmax block

by Lemma 1. For each case, we can determine a constant C_{k+1} that satisfies $(E^\alpha_{A_{k+1}} \circ \cdots \circ E^\alpha_{A_0})(0) \le E^\alpha_{A_{k+1}}(C_k 2^\alpha) \le C_{k+1}2^{-\alpha}$. Thus, the theorem is proved.

APPENDIX F
GENERALIZATION OF THEOREM 4 FOR ResNet MODEL

ResNet model cannot be decomposed of basic blocks, since it contains a "residual block". The authors of [14] suggest an operation R that satisfies

$$R(\mathbf{x}) = \mathcal{G}(\mathbf{x}) + \mathbf{Px},$$

where $\mathcal{G}(\mathbf{x})$ denotes the residual mapping which will be learned, and $\mathbf{P}$ is a linear projection matrix to match the dimension of $\mathcal{G}(\mathbf{x})$. In this section, we call such operation R as a *residual block*. In the residual block designed in ResNet, all residual mapping $\mathcal{G}(\mathbf{x})$ is a composition of basic blocks [14]. Therefore, by Theorem 4, we can determine a constant $C_\mathcal{G}$ that satisfies $\|\mathcal{G}^\alpha(\mathbf{x}) - \mathcal{G}(\mathbf{x})\|_\infty \le C_\mathcal{G}2^{-\alpha}$. In case of $\mathbf{P}$, there are two methods of constructing projection in ResNet: the first one pads extra zeros, and the second one uses 1×1 convolution [14]. We note that both methods can be considered as linear blocks, and thus the approximate block $R^\alpha(\mathbf{x})$ can be represented by $\mathcal{G}^\alpha(\mathbf{x}) + \mathbf{Px}$. Considering this situation, we generalize the original Theorem 4 so that it is also valid for ResNet model.

Theorem 5 (Generalized version of Theorem 4): For $\alpha \ge 4$ and a basic deep learning model $\mathcal{F}$, there exists a constant C such that $\|\mathcal{F}^\alpha(\mathbf{x}) - \mathcal{F}(\mathbf{x})\|_\infty \le C2^{-\alpha}$ for every input $\mathbf{x}$, where the constant C can be determined only by model parameters.

Proof: We prove this statement by obtaining error propagation function $E^\alpha_R(e)$ of residual block $R(\mathbf{x}) = \mathcal{G}(\mathbf{x}) + \mathbf{Px}$. From the definition of error propagation function, we have

$$\begin{aligned}E^\alpha_R(e) &= \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty\le B,\,\|\mathbf{e}\|_\infty\le e} \|R^\alpha(\mathbf{x}+\mathbf{e}) - R(\mathbf{x})\|_\infty \\ &\le \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty\le B,\,\|\mathbf{e}\|_\infty\le e} (\|R^\alpha(\mathbf{x}+\mathbf{e}) - R(\mathbf{x}+\mathbf{e})\|_\infty \\ &\quad + \|R(\mathbf{x}+\mathbf{e}) - R(\mathbf{x})\|_\infty) \\ &\le \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty\le B,\,\|\mathbf{e}\|_\infty\le e} (\|\mathcal{G}^\alpha(\mathbf{x}+\mathbf{e}) - \mathcal{G}(\mathbf{x}+\mathbf{e})\|_\infty \\ &\quad + \|\mathcal{G}(\mathbf{x}+\mathbf{e}) - \mathcal{G}(\mathbf{x}) + \mathbf{Pe}\|_\infty) \\ &\le C_\mathcal{G}2^{-\alpha} + \|\mathbf{P}\|_\infty e \\ &\quad + \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty\le B,\,\|\mathbf{e}\|_\infty\le e} \|\mathcal{G}(\mathbf{x}+\mathbf{e}) - \mathcal{G}(\mathbf{x})\|_\infty,\end{aligned}$$

where $C_\mathcal{G}$ is the constant that satisfies $\|\mathcal{G}^\alpha(\mathbf{x}) - \mathcal{G}(\mathbf{x})\|_\infty \le C_\mathcal{G}2^{-\alpha}$. To estimate $\|\mathcal{G}(\mathbf{x}+\mathbf{e}) - \mathcal{G}(\mathbf{x})\|_\infty$, we decompose residual mapping $\mathcal{G}$ into $G_k \circ \cdots \circ G_0$, where G_i's are basic blocks. At first, we define

$$\Delta_A(e) = \sup_{\|\mathbf{x}+\mathbf{e}\|_\infty\le B,\,\|\mathbf{e}\|_\infty\le e} \|A(\mathbf{x}+\mathbf{e}) - A(\mathbf{x})\|_\infty$$

for a block A and magnitude of the error, e. (Note that the definitions of $\Delta_A(e)$ and $E^\alpha_A(e)$ are different.) Then $\Delta_A(e)$ is a non-decreasing function of e for every block A. Thus, similar argument in the proof of Theorem 3 shows that

$$\|\mathcal{G}(\mathbf{x}+\mathbf{e}) - \mathcal{G}(\mathbf{x})\|_\infty \le (\Delta_{G_k} \circ \cdots \Delta_{G_0})(\|\mathbf{e}\|_\infty) \tag{14}$$

for every error vector $\mathbf{e}$. Also, similar argument in the proof of Lemma 1 shows that

$$\Delta_A(e) \le \begin{cases} \|\mathbf{A}\|_\infty e, & A : \text{linear block, } A(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \\ e, & A : \text{ReLU} \\ e, & A : \text{max-pooling} \\ e/2 & A : \text{softmax} \end{cases} \tag{15}$$

for every basic block A and $e \ge 0$. (These inequalities for $\Delta_A(e)$ corresponds to the inequalities for $E_A^\alpha(e)$ in Lemma 1 when the precision parameter α goes to infinity.) From two inequalities (14) and (15), there exists a constant $C'_{\mathcal{G}}$ that satisfies $\|\mathcal{G}(\mathbf{x}+\mathbf{e}) - \mathcal{G}(\mathbf{x})\|_\infty \le C'_{\mathcal{G}}\|\mathbf{e}\|_\infty$ since G_i's are all basic blocks. Therefore, we have

$$E_R^\alpha(e) \le C_{\mathcal{G}} 2^{-\alpha} + (\|\mathbf{P}\|_\infty + C'_{\mathcal{G}})e$$

for every $e \ge 0$.

Now, we prove the theorem using mathematical induction. Let $\mathcal{F}$ be a deep learning model which can be decomposed into $A_n \circ \cdots \circ A_0$ where A_k's are all basic blocks or residual blocks. First, for $n = 0$, assume that $\mathcal{F}$ has one block A. The upper bounds of $E_A^\alpha(0)$ for the four basic blocks and residual blocks have forms of $C_0 2^{-\alpha}$, where C_0 can be zero. Inductively, for $n = k$, $(E_{A_k}^\alpha \circ \cdots \circ E_{A_0}^\alpha)(0) \le C_k 2^{-\alpha}$ for some C_k. For $n = k+1$, we can determine a constant C_{k+1} that satisfies $(E_{A_{k+1}}^\alpha \circ \cdots \circ E_{A_0}^\alpha)(0) \le C_{k+1}2^{-\alpha}$ when A_{k+1} is a basic block (see the proof of Theorem 4). If A_{k+1} is a residual block with $A_{k+1}(\mathbf{x}) = \mathcal{G}(\mathbf{x}) + \mathbf{P}\mathbf{x}$ for some residual mapping $\mathcal{G}(\mathbf{x})$ and linear projection matrix $\mathbf{P}$, then

$$E_{A_{k+1}}^\alpha(C_k 2^\alpha) \le C_{\mathcal{G}} 2^{-\alpha} + (\|\mathbf{P}\|_\infty + C'_{\mathcal{G}})(C_k 2^\alpha) = C_{k+1} 2^{-\alpha}$$

where $C_{k+1} = C_{\mathcal{G}} + (\|\mathbf{P}\|_\infty + C'_{\mathcal{G}})C_k$ which is independent of α. Therefore,

$$(E_{A_{k+1}}^\alpha \circ \cdots \circ E_{A_0}^\alpha)(0) \le E_{A_{k+1}}^\alpha(C_k 2^\alpha) \le C_{k+1}2^{-\alpha},$$

which completes the proof. ■

REFERENCES

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

[2] B. Reagen, W. Choi, Y. Ko, V. T. Lee, H. S. Lee, G. Wei, and D. Brooks, "Cheetah: Optimizing and accelerating homomorphic encryption for private inference," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2021, pp. 26–39.

[3] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, "NGraph-HE: A graph compiler for deep learning on homomorphically encrypted data," in *Proc. 16th ACM Int. Conf. Comput. Frontiers*, Apr. 2019, pp. 3–13.

[4] 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.

[5] 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. (ICML)*, 2016, pp. 201–210.

[6] T. van Elsloo, G. Patrini, and H. Ivey-Law, "SEALion: A framework for neural network inference on encrypted data," 2019, *arXiv:1904.12840*.

[7] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei, "Faster CryptoNets: Leveraging sparsity for real-world encrypted inference," 2018, *arXiv:1811.09953*.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[9] A. Krizhevsky. (2009). *Learning Multiple Layers of Features From Tiny Images*. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[10] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, "High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Cham, Switzerland: Springer, 2021, pp. 618–647.

[11] Y. Lee, J. Lee, Y.-S. Kim, H. Kang, and J.-S. No, "High-precision bootstrapping for approximate homomorphic encryption by error variance minimization," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, Norway. Cham, Switzerland: Springer, 2022, pp. 551–580.

[12] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Cham, Switzerland: Springer, 2021, pp. 587–617.

[13] 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, no. 4, pp. 114–148, Aug. 2021.

[14] 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.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–9.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[17] A. Al Badawi, J. Chao, J. Lin, C. F. Mun, J. J. Sim, B. H. M. Tan, X. Nan, K. M. M. Aung, and V. R. Chandrasekhar, "Towards the AlexNet moment for homomorphic encryption: HCNN, theFirst homomorphic CNN on encrypted data with GPUs," 2018, *arXiv:1811.00778*.

[18] P. Xie, B. Wu, and G. Sun, "BAYHENN: Combining Bayesian deep learning and homomorphic encryption for secure DNN inference," 2019, *arXiv:1906.00639*.

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

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

[21] C.-C. Wang, C.-H. Tu, M.-C. Kao, and S.-H. Hung, "TensorHE: A homomorphic encryption transformer for privacy-preserving deep learning," in *Proc. Conf. Res. Adapt. Convergent Syst.*, Oct. 2022, pp. 124–130.

[22] 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.

[23] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, pp. 34–91, Jan. 2020.

[24] Q. Lou and L. Jiang, "She: A fast and accurate deep neural network for encrypted data," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 10035–10043.

[25] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade, "TAPAS: Tricks to accelerate (encrypted) prediction as a service," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 4490–4499.

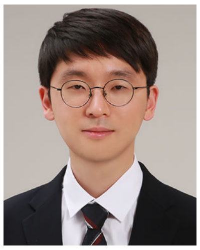
[26] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Proc. Annu. Int. Cryptol. Conf. (CRYPTO)*. Cham, Switzerland: Springer, 2018, pp. 483–512.

[27] S. Meftah, B. H. M. Tan, C. F. Mun, K. M. M. Aung, B. Veeravalli, and V. Chandrasekhar, "DOReN: Toward efficient deep convolutional neural networks with fully homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3740–3752, 2021.

[28] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security (ASIACRYPT)*. Cham, Switzerland: Springer, 2017, pp. 409–437.

[29] 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," 2021, *arXiv:2106.07229*.
[30] 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 Cryptography*. Cham, Switzerland: Springer, 2018, pp. 347–368.
[31] 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 convolutions," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 12403–12422.
[32] E. Lee, J. Lee, J. No, and Y. Kim, "Minimax approximation of sign function by composite polynomial for homomorphic comparison," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 3711–3727, Nov. 2022.
[33] J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security (ASIACRYPT)*. Cham, Switzerland: Springer, 2020, pp. 221–256.
[34] E. Y. Remez, "Sur La détermination des polynômes d'approximation de degré donnée," *Commun. Soc. Math. Kharkov*, vol. 10, no. 196, pp. 41–63, 1934.
[35] M. S. Paterson and L. J. Stockmeyer, "On the number of nonscalar multiplications necessary to evaluate polynomials," *SIAM J. Comput.*, vol. 2, no. 1, pp. 60–66, Mar. 1973.
[36] J.-W. Lee, E. Lee, Y.-W. Lee, and J.-S. No, "Optimal minimax polynomial approximation of modular reduction for bootstrapping of approximate homomorphic encryption," Cryptol. ePrint Arch., Bellevue, WA, USA, Tech. Rep. 2020/552, 2nd Version, 2020. [Online]. Available: https://eprint.iacr.org/2020/552/20200803:084202
[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[38] *Microsoft SEAL (Release 3.7)*. Microsoft Res., Redmond, WA, USA, Sep. 2021. [Online]. Available: https://github.com/Microsoft/SEAL

JUNGHYUN LEE (Graduate Student Member, IEEE) received the B.S. degree in statistics and the M.S. degree in mathematics from Seoul National University, Seoul, South Korea, in 2018 and 2021, respectively, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His current research interests include homomorphic encryption and lattice-based cryptography.



EUNSANG LEE received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2014 and 2020, respectively. From 2020 to 2022, he was a Postdoctoral Researcher with the Institute of New Media and Communications, Seoul National University. He is currently an Assistant Professor with the Department of Software, Sejong University, Seoul.

JOON-WOO LEE (Member, IEEE) received the B.S. and Ph.D. degrees in electrical and computer engineering with Seoul National University, Seoul, South Korea, in 2016 and 2022, respectively. He is currently an Assistant Professor with the Engineering Department, School of Computer Science, Chung-Ang University, Seoul. His research interests include privacy-preserving machine learning, homomorphic encryption, and post-quantum cryptography.

YONGJUNE KIM (Member, IEEE) received the B.S. (Hons.) and M.S. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2002 and 2004, respectively, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2016. From 2016 to 2018, he was a Postdoctoral Scholar with the Coordinated Science Laboratory, University of Illinois at Urbana–Champaign, Urbana, IL, USA. From 2018 to 2020, he was a Researcher with Western Digital Research, Milpitas, CA, USA. From 2020 to 2022, he was an Assistant Professor with the Department of Electrical Engineering and Computer Science, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, South Korea. Since 2022, he has been with the Department of Electrical and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, South Korea, where he is currently an Assistant Professor. His research interests include machine learning, coding theory, and information theory. He received the IEEE Data Storage Best Student Paper Award, the Best Paper Award of the 2016 IEEE International Conference on Communications (ICC), the Best Paper Award (honorable mention) of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), and the Best Paper Award of the 31st Samsung Semiconductor Technology Symposium.

YOUNG-SIK KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, in 2001, 2003, and 2007, respectively. He joined the Semiconductor Division, Samsung Electronics, where he worked in the research and development of security hardware IPs for various embedded systems, including modular exponentiation hardware accelerator (called Tornado 2MX2) for RSA and elliptic-curve cryptography in smart-card products and mobile application processors of Samsung Electronics, until 2010. He is currently a Professor with Chosun University, Gwangju, South Korea. He is also a Submitter for two candidate algorithms (McNie and pqsigRM) in the first round for the NIST Post Quantum Cryptography Standardization. His research interests include post-quantum cryptography, the IoT security, physical-layer security, data hiding, channel coding, and signal design. He is selected as one of 2025's 100 Best Technology Leaders (for Crypto-Systems) by the National Academy of Engineering, South Korea.

JONG-SEON NO (Fellow, IEEE) received the B.S. and M.S.E.E. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1981 and 1984, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1988. He was a Senior MTS with Hughes Network Systems, from 1988 to 1990. He was an Associate Professor with the Department of Electronic Engineering, Konkuk University, Seoul, from 1990 to 1999. He joined the Faculty of the Department of Electrical and Computer Engineering, Seoul National University, in 1999, where he is currently a Professor. His research interests include error-correcting codes, cryptography, sequences, LDPC codes, interference alignment, and wireless communication systems. He became an IEEE Fellow through the IEEE Information Theory Society, in 2012. He became a member of the National Academy of Engineering of Korea (NAEK), in 2015, where he served as the Division Chair for Electrical, Electronic, and Information Engineering, from 2019 to 2020. He was a recipient of the IEEE Information Theory Society Chapter of the Year Award, in 2007. From 1996 to 2008, he served as the Founding Chair for the Seoul Chapter of the IEEE Information Theory Society. He was the General Chair of Sequence and Their Applications, in 2004 (SETA 2004), Seoul. He also served as the General Co-Chair for the International Symposium on Information Theory and its Applications, in 2006 (ISITA 2006), and the International Symposium on Information Theory, in 2009 (ISIT 2009), Seoul. He served as the Co-Editor-in-Chief for the IEEE Journal of Communications and Networks, from 2012 to 2013.

• • •