

RESEARCH ARTICLE

Toward Bootstrapping-Free Homomorphic Encryption-Based GRU Network for Text Classification

ZEYU WANG¹ AND MAKOTO IKEDA¹, (Senior Member, IEEE)

Department of Electrical Engineering and Information Systems, The University of Tokyo, Tokyo 113-0032, Japan

Corresponding author: Zeyu Wang (wangzy@silicon.u-tokyo.ac.jp)

ABSTRACT Homomorphic encryption (HE) is a promising method in privacy-preserving cloud computing. Applying HE on feedforward neural networks has been frequently reported recently but the research on recurrent neural networks is still insufficient. In previous studies, HE-based GRU is built with bootstrapping due to the changeable input length and large number of required multiplications, which is not time-efficient. In this study, we give a guideline of building bootstrapping-free HE-based GRU for text classification tasks. We discuss the methods of pre-processing of texts to decrease the input sequence length but keep the accuracy in a comparable level as the original GRU. The architecture of GRU is designed with flexibility to process the input sequence with different lengths while fixing the number of recurrent steps. At last, the HE parameter selection is discussed. We analyze the noise raised from HE operations and select the parameters that ensure the results from encrypted data are the same as that on plaintexts. The proposed model is evaluated on 6 popular text datasets, and the results show that the accuracy is only lower than the original GRU by at most 4.2%. Despite the complicated calculations in GRU, the proposed model is light-weighted and the fastest inference among our implementation costs only 10 minutes. We show the potential of applying HE schemes on complex models without bootstrapping to achieve fast encrypted computations.

INDEX TERMS CKKS, gated recurrent unit, homomorphic encryption, privacy-preserving technique.

I. INTRODUCTION

During past decades, deep learning has been applied across a wide range of scenarios. While the deep learning models are becoming powerful and precious, their complex architecture and heavy computation cost make it hard to run on personal computers. The development of cloud computing, where users upload data and cloud server performs the deep learning computations, enables more people to enjoy the benefits of deep learning. However, in some scenarios, such as genetic analysis, smart city and financial forecasting, user data is sensitive and cannot be exposed to external servers. Besides, the parameters of a well-trained deep learning model may also be protected by the model provider. Several privacy-preserving cloud computing schemes have been proposed to address concerns regarding data privacy, such as Virtual Machine (VM), Multi-Party Computation (MPC)

The associate editor coordinating the review of this manuscript and approving it for publication was Gang Li¹.

and Homomorphic Encryption (HE) [1]. In VM, a part of the server is isolated and allocated to one user. The user can only access the data on the allocated part, but it does not block the attacks from untrusted servers or through internet communication. MPC is to complete one computation by several independent parties. As long as the private communication among these parties is not allowed, the data is never exposed to other parties. However, this method relies on stable communications among participants, making computing speed highly dependent on internet speed. In HE schemes, additions and multiplications on ciphertexts are supported, which enables encrypted calculations. Compared to the two methods above, HE is a promising solution because of its high security and relaxed communication requirements.

A. HOMOMORPHIC ENCRYPTION

HE is a cryptographic solution that protects the data privacy by encrypting all data, including that from cloud service

users and deep learning model providers, into ciphertexts. All the calculations as well as the results are encrypted and the security is guaranteed by the HE scheme. Besides, during 1 deep learning computation, only 2 communications between user and server are required: one for uploading the encrypted data and another for downloading the encrypted results. However, the largest drawback of HE is its slow speed. Typically, computations on ciphertexts are slower than those on plaintexts by 3-4 orders of magnitude. Therefore, developing efficient HE algorithms and HE-based calculating schemes has attracted much research attention.

Since the first lattice-based HE algorithms proposed in [2], faster and more efficient HE algorithms, including BGV, BFV and CKKS, are developed and utilized in privacy-preserving deep learning applications [3], [4], [5]. Among these algorithms, CKKS (Cheon-Kim-Kim-Song) algorithm shows the advantage for deep learning applications because the encryption is designed for floating-point numbers, which can be directly applied on calculations. In this paper, the privacy-preserving deep learning model is designed on CKKS algorithm.

B. HE-BASED DEEP LEARNING

The first attempt of HE-based neural network (NN) is proposed in [6]. In recent years, HE is applied on more complex and deeper NN structures, such as Convolutional Neural Networks (CNNs) [7], [8], [9], [10], [11], [12]. However, we notice that most implementations focus on Feedforward Neural Networks (FNNs), while the HE-related research on another widely used architecture, Recurrent Neural Networks (RNNs), is still insufficient. As a result, the evaluations of HE-based NNs are almost on image recognition tasks, and only limited literature reports the HE applications on time series processing.

Compared to FNNs, there are several difficulties when applying HE schemes on RNNs. First, as the number of supported multiplications is decided by HE parameters, dealing with the changeable input sequence lengths poses an challenge in the design of HE-based RNNs. In FNNs, the number of multiplications remains constant for a given model, making it simple to determine HE parameters. However, the number of multiplications needed for an RNN inference depends on the lengths of input time series. To support the inference on potential long input sequences, large HE parameters must be set, which leads to slow computing speed. While bootstrapping, an HE operation to refresh the ciphertext, can be used to increase the number of supported multiplications, its slow speed always makes HE-based applications impractical [8], [14]. Second, RNNs are less robust to computing errors compared to FNNs. Since only additions and multiplications are supported in HE schemes, non-linear functions in RNNs are substituted with polynomials in HE-based NNs, which introduces error into calculations. Besides, the precision of HE-based calculations may degrade depending on the selected HE parameters. These calculating errors may accumulate

through RNN inference, resulting in a low accuracy performance.

There are some research focusing on HE-based RNNs [15], [16]. However, the implementation is on simple RNN structure, which is reported as inadequate for completing complex tasks. Gated Recurrent Unit (GRU) is a popular variant of RNNs showing improved performance, particularly when handling long input sequences [13]. Nevertheless, its complex architecture increases the difficulty on building with HE. We find that the previous designs of HE-based GRU models mainly focus on the implementing techniques at HE scheme side, while the discussion on GRU architecture is limited [14], [17]. In this paper, we present several techniques for designing the GRU architecture to be friendly with HE, including pre-processing techniques, GRU core design and HE parameter selection strategies to achieve efficient and accurate privacy-preserving GRU inference.

C. OUR CONTRIBUTIONS

This study proposes a guideline for building bootstrapping-free HE-based GRUs. We point out that bootstrapping is one of the main bottlenecks in improving the speed of the HE-based GRU inferences. We avoid bootstrapping by employing pre-processing techniques and adjusting the GRU architecture. The number of required multiplications during a GRU inference is significantly reduced, while the long input sequences are still supported. The proposed model is evaluated on text classification tasks, which is one of the main GRU applications. The selected text datasets encompass a wide range of lengths, from short sentences to long paragraphs. The key contributions of this study are summarized below:

- 1) We provide a guideline of building HE-friendly GRU architectures for text classification tasks. We show the effectiveness of the proposed methods on the text datasets in different scales and classifying difficulties.
- 2) The pre-processing of texts is discussed in this study. Different from setting a common parameter, we optimize pre-processing procedure for each dataset. The necessary information for inference is remained and the accuracy is preserved.
- 3) We discuss the HE parameter selection and analyze the noise introduced by HE-based calculations. An optimized scaling factor is selected to support more multiplications while maintaining precision equalling to that on plaintexts.
- 4) We demonstrate a lightweight HE-based GRU implementation to achieve fast encrypted inference. Despite the complex calculations in GRU, the running time is reduced compared to other HE-based RNN designs.

II. RELATED WORK

Techniques for applying HE on NNs are frequently reported in recent years. These techniques include implementations of deep NNs [8], [19], accelerated bootstrapping [7], [20], [21], and hardware accelerations [22], [23], [24], [25],

TABLE 1. Overview of HE-based RNN implementations.

| | Year | RNN type | Dataset | Task | Bootstrapping-free |
|------|------|------------|---|---|--------------------|
| [15] | 2021 | Simple RNN | AmazonReview, IMDB | Text classification | ✓ |
| [17] | 2021 | GRU | EnronEmails, PennTreebank, IMDB, YelpReview | Text classification | × |
| [14] | 2022 | GRU | MNIST, deepTarget | Sequence modeling, regression, classification | × |
| [18] | 2023 | GRU | MNIST, AG_NEWS | Text/image classification | ✓ |

[26], [27]. In [8], a very deep model is implemented using HE, demonstrating the feasibility of HE on large-scale NNs. However, bootstrapping is frequently performed to support the large number of required multiplications, which consumes 31.5% of the computing time. As a result, the latency is very high that only one inference costs about 3 hours, which is not practical for real-world applications.

Compared to HE-based FNNs, there are fewer designs on RNNs. In [17], an HE-based GRU is developed for text classification and regression tasks. However, in addition to HE schemes, garbled circuits (GCs) are utilized to achieve encrypted activation functions, which requires a large amount of communications between client and server before calculations. Despite this, bootstrapping is still required to support long recurrent steps. Authors in [15] implement an RNN model using only HE schemes and reduce the recurrent steps by a similar splitting method described in this paper. However, the implementation is based on the simple RNN structure and suffers from low accuracy on complex tasks. In [14], the authors design an HE-based GRU with high-precision approximation of the original model. High-order polynomials are used to replace non-linear activation functions in the original GRU. However, bootstrapping is frequently performed to support the multiplications in evaluating polynomials, which results in slow speed. We compare the previous work in Tab. 1 to provide an overview of HE-based RNN implementations.

In our previous work, a bootstrapping-free HE-based GRU structure is proposed [18]. By modifying the original GRU architecture, we significantly reduce the number of required multiplications. Two key techniques are introduced: rearrangement of input sequences and applying layer normalization before activation. The proposed model is evaluated on MNIST and AG_NEWS datasets, on which image recognition and text classification tasks are performed, respectively. Additionally, a ciphertext packing technique is employed to enable encrypted inference performing in a Single-Instruction-Multiple-Data (SIMD) way. As a result, the speed and throughput are greatly improved compared to [14]. However, MNIST and AG_NEWS datasets consist of sequences with short lengths, and the evaluation on more complex texts is lacking. In this paper, we propose new techniques in HE-based GRU design and present evaluations on large-scale text sequences. We also analyze the calculating errors through encrypted GRU inferences and discuss HE parameter selection.

III. PRELIMINARIES

A. CKKS SCHEME

CKKS, as well as other popular HE schemes such as BGV and BFV, relies on a NP-hard problem known as ring learning with error (RLWE). The basic concept is to introduce slight noise in encryption to make it hard to decrypt without the secret key. However, the noise in each ciphertext accumulates through HE operations, particularly multiplications. A decryption failure happens when the noise reaches a threshold value. The maximum multiplicative depth, indicating the number of maximum supported multiplications, is determined by the selected HE parameter set.

CKKS is designed for the encryption of floating-point numbers [5]. Compared to other HE schemes designed on integers, CKKS shows several advantages for NN applications. Firstly, the procedure of converting floating-point numbers to integers before encryption is avoided in CKKS-based implementations. In other HE schemes, this conversion is typically achieved by multiplying the data with a constant scaling factor. However, through calculations, the scaling factor accumulates and must be carefully controlled to prevent overflow. This step introduces extra design in HE-based NNs and increases computational difficulty. Secondly, CKKS supports approximate encryption. Through calculations in CKKS, the increasing noise in a ciphertext gradually degrades the computing precision. In contrast, the increasing noise results in a total decryption failure when exceeding a threshold noise level in other HE schemes. To prevent this, large HE parameters must be set to increase the noise tolerance range. In CKKS, smaller HE parameters can be used compared to other HE schemes, which enhances the computing speed.

The encryption/decryption and HE operations in CKKS are summarized as below. For a positive integer M , let $\Phi_M(X)$ be the M -th cyclotomic polynomial of degree $N = \Phi(M)$. Let $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$ be the ring of integers of a number field $\mathbb{Q}[X]/(\Phi_M(X))$. $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ is the residue ring of \mathcal{R} modulo an integer q .

- $\text{Encode}(\mathbf{z}; \Delta)$: Encode the vector $\mathbf{z} \in \mathbb{C}^{N/2}$ to a polynomial $m(X) \in \mathcal{R}$ for encryption. Δ is the scaling factor to preserve the accuracy in HE operations.
- $\text{KeyGen}(1^\lambda)$: Generate the secret key sk for decryption, the public key pk for encryption and ensure the security level as λ .
- $\text{Enc}(m; pk)$: Encrypt the plaintext message m into a ciphertext $\mathbf{c} \in \mathcal{R}_{q_L}^k$, where k is a fixed integer and q_L is

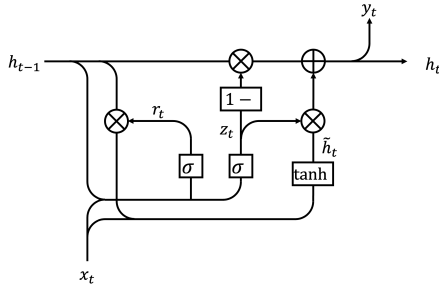


FIGURE 1. Architecture of a GRU cell.

the modulus of ciphertext decided by the multiplicative depth L .

- Dec($\mathbf{c}; sk$): Decrypt the ciphertext \mathbf{c} into plaintext message $m \in \mathcal{R}$ using secret key sk .
- Decode(m): Decode the plaintext message $m \in \mathcal{R}$ into a vector $\mathbf{z} \in \mathbb{C}^{N/2}$.
- Add($\mathbf{c}_1, \mathbf{c}_2$): Let \mathbf{c}_1 and \mathbf{c}_2 be the encryptions of the plaintexts messages m_1 and m_2 with the same scaling factor. Output the encryption of $m_1 + m_2$.
- Mult($\mathbf{c}_1, \mathbf{c}_2$): Let \mathbf{c}_1 and \mathbf{c}_2 be the encryption of the plaintexts messages m_1 and m_2 with rescaling factors Δ_1 and Δ_2 . Output the encryption of $m_1 m_2$ with rescaling factor of $\Delta_1 \Delta_2$.
- LeftRotate(\mathbf{c}, k): Let the corresponding plaintext message be $m(X) = a_0 + a_1 X + \dots + a_{N-1} X^{N-1}$. Rotate the coefficients to the left side and output the encryption of $m'(X) = a_k + a_{k+1} X + \dots + a_{k-1} X^{N-1}$. k is an integer and $k > 0$.
- RightRotate(\mathbf{c}, k): Similar to LeftRotate but rotate the coefficients to the right side.
- Rescaling(\mathbf{c}, δ): Rescale the modulus q and scaling factor Δ of ciphertext \mathbf{c} to q/δ and Δ/δ , respectively.

B. GRU NETWORK

GRU is a variant of RNNs, with enhanced ability to process long-term dependencies in input sequences compared to the original RNNs. Compared to another popular variant, LSTM, GRU has a simplified structure while maintaining a comparable performance as LSTM, which is important in HE schemes for reducing computational complexity and increasing speed. The structure of one GRU cell is illustrated in Fig 1, with the calculations expressed as follows:

$$\begin{aligned} r_t &= \sigma(W_{rh}h_{t-1} + W_{rx}x_t + b_r), \\ z_t &= \sigma(W_{zh}h_{t-1} + W_{zx}x_t + b_z), \\ \tilde{h}_t &= \tanh(W_{hh}(r_t \cdot h_{t-1}) + W_{hx}x_t + b_z), \\ h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t, \end{aligned} \quad (1)$$

where σ is the Sigmoid function. The output of a GRU network is derived by a linear layer

$$y_L = W_{yh}h_L + b_y, \quad (2)$$

where L is the length of the input sequence.

To perform text classification tasks, first the input texts are converted into numbers for GRU computation. This

TABLE 2. Datasets used for GRU evaluation.

| Dataset | Average length | Classes |
|--------------------------------|----------------|---------|
| AG_NEWS [28] | 43 | 4 |
| DBpedia [29] | 54 | 14 |
| AmazonReview ¹ [28] | 92 | 2 |
| YahooAnswers [28] | 110 | 10 |
| YelpReview ² [28] | 157 | 2 |
| IMDB [30] | 270 | 2 |

^{1,2} Precisely the AmazonReviewPolar, YelpReviewPolar datasets are used here, where the samples are labeled in 2 classes.

conversion includes 2 steps: translation and embedding. Translation is to map each word, phrase and punctuation mark to a unique number based on a pre-established vocabulary, which derives an 1-d sequence. Then embedding is performed to project each element in the 1-d sequence to a vector with a fixed length known as the embedding size. This projection is achieved by a matrix multiplication, where the matrix is also trained during GRU training.

IV. HE-BASED GRU DESIGN

The total available multiplications for one ciphertext are determined by HE parameters. Without bootstrapping, larger HE parameters allow for more multiplications but also result in significantly higher computational costs. To make the HE-based GRU practical in both accuracy performance and speed, we have 2 strategies: 1) to reduce the required multiplications in a GRU inference and 2) to optimize HE parameters to increase the supported multiplications. For the first strategy, let N_{mul} denote the number of total multiplications in one inference, then

$$N_{mul} = N_{re} \times L + N_{lin}, \quad (3)$$

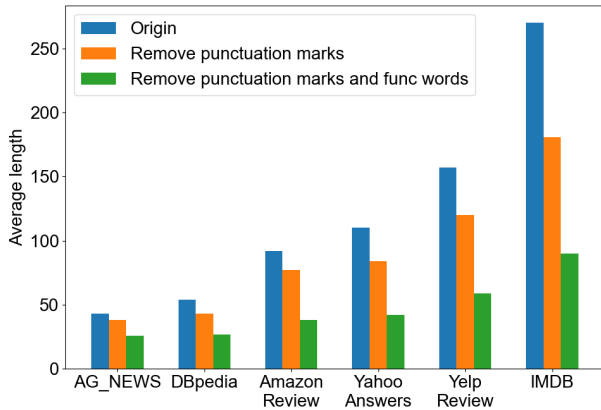
where N_{re} and N_{lin} represent the number of multiplications required in one recurrent step and one linear layer (output layer), respectively. L denotes the number of recurrent steps, which is depended by the length of input sequences. From Equ. 3 we can know that the required multiplications in one inference is mainly decided by N_{re} and L . In Sec. IV-A and Sec. IV-B, we discuss the method to decrease L and in Sec. IV-C we show how to decrease N_{re} . The second strategy of optimizing HE parameters is addressed in Sec. IV-D.

We evaluate the text classification tasks on 6 text datasets containing English sentences (Tab. 2). The complexity of classification task varies in datasets due to the differences in sentence length and number of classes. For the GRU parameters, we set embedding size and hidden size to 64 in the following experiments, which is a typical setting in practical GRU applications. The experiments for GRU architecture design are performed on plaintexts, and in Sec. IV-D we demonstrate that the accuracy on ciphertexts is kept the same as that on plaintexts.

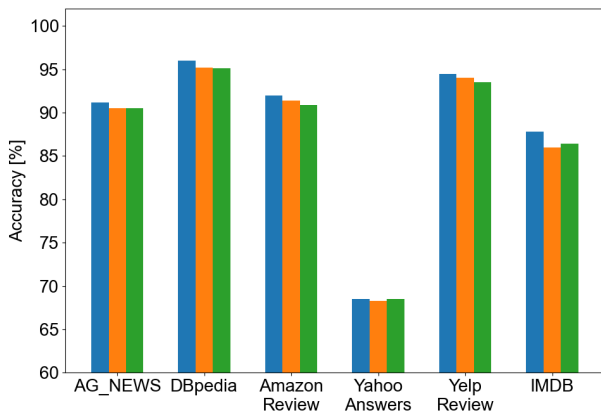
A. PREPROCESSING OF TEXTS

1) REMOVE OF FUNCTION WORDS AND PUNCTUATION MARKS

In an English sentence, words can be categorized into content words, which include nouns, verbs, adjectives, and adverbs,



(a) Average text length before and after removing punctuation marks / punctuation marks and function words.



(b) GRU accuracy before and after removing punctuation marks / punctuation marks and function words.

FIGURE 2. Average text length and GRU accuracy before and after removing punctuation marks / punctuation marks and function words. After removing the input text length is highly decreased while accuracy is preserved.

as well as function words, such as prepositions, pronouns, and conjunctions. From human's perspective, we believe that content words play a more important role than most function words in classifying a paragraph based on its meaning. If using only content words is sufficient in GRU inference, the input sequence length can be greatly reduced. However, in the tasks involving people's reviews or emotions, such as classifying texts in AmazonReview, YelpReview and IMDB datasets, function words with negative meanings such as 'no,' 'not,' and 'nothing' may totally alter the sentence's meaning. In our design, we remain the function words with negative meaning but remove the others. Furthermore, in original GRUs, punctuation marks are usually embedded as other words and used for inferences. Similarly to function words, punctuation marks lack inherent meaning but help readers to segment and comprehend the text. Therefore, we also test removing punctuation marks to decrease the input text length.

The average lengths of the texts in each dataset before and after removing punctuation marks, or both function words

and punctuation marks are shown in Fig. 2. After removing both function words and punctuation marks, the length is significantly reduced by 40%-66%, especially for the dataset with long texts. This preprocessing step highly reduces the difficulty of processing long sequences.

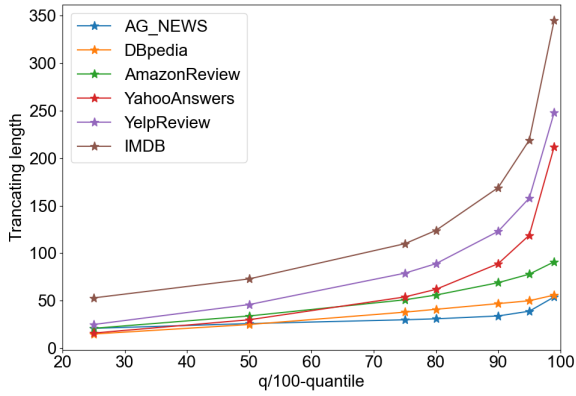
We evaluate each dataset in 3 conditions: without removal, only removing punctuation marks, and removing both punctuation marks and function words. The results are shown in Fig. 2b. It can be observed that there is no obvious difference in accuracy before and after removing punctuation marks or both punctuation marks and function words. Even though after removal, the sentence structures are not preserved and become difficult for humans to understand, content words and function words with negative meanings still contain essential messages for GRU to provide accurate classification results. The sacrifice in accuracy after removal is only 2.9%, which is acceptable considering the decrease in input text length. Through this method, the required recurrent steps are significantly reduced while the accuracy is almost preserved.

2) EARLY TRUNCATION

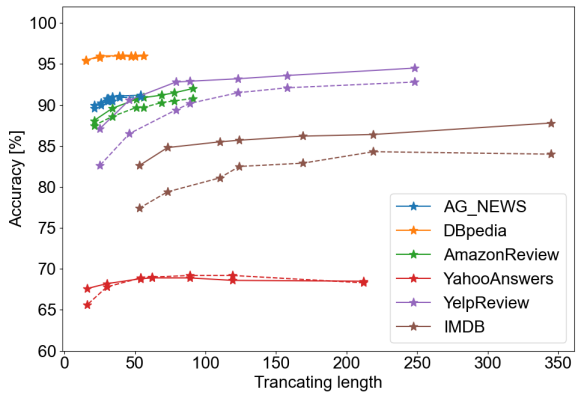
In addition to removing punctuation marks and function words, early truncation is another method to reduce the input text length. We believe that for some long texts, the information contained in the first several words may be sufficient for GRU classification. Different from simply truncating at fixed lengths, in our experiments the texts are truncated at specific lengths determined by the statistical features of each dataset to find the optimized value. The statically features are derived after removing punctuation marks and function words (Sec. IV-A1), and the truncation is also applied on the shortened sentences.

Fig. 3 illustrates the truncating lengths used in our experiments for each dataset. These lengths are determined by $q/100$ -quantiles, where q is set to (99, 95, 90, 80, 75, 50, 25). The GRU is evaluated on each dataset with each truncating length, and the results are shown in the solid lines in Fig. 3b and Fig. 3c (solid lines in the 2 figures are identical). It can be observed that with the truncating length decreasing, there is only slight accuracy degradation until the input text becomes too short. Specifically, truncating at the 75/100-quantile only degrades the accuracy within 2% for each dataset. Especially for AG_NEWS, DBpedia and YahooAnswers datasets, even truncating at the 25/100-quantiles is sufficient to keep the accuracy degradation around 1%. From Fig. 3, we know that truncating at 75/100-quantiles or shorter lengths obviously decreases the text length. Particularly for the datasets containing long texts, such as IMDB, YelpReview, and YahooAnswers, truncating at the 75/100-quantiles reduces the length by around 70%, thereby significantly reducing the input sequence length to GRU.

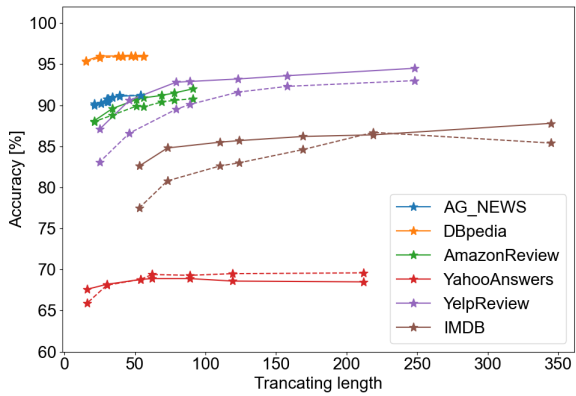
The specific truncating lengths for each dataset are not determined here but discussed in Sec. IV-B. Optimized truncating lengths for each dataset are selected based on the experiments of rearranging input sequences.



(a) Truncating length at each $q/100$ -quantile.



(b) GRU accuracy at each truncating length. Solid lines show the result without input rearrangement and dashed lines show the result with input rearrangement when recurrent step $L = 5$.



(c) GRU accuracy at each truncating length. Solid lines show the result without input rearrangement and dashed lines show the result with input rearrangement when recurrent step $L = 11$.

FIGURE 3. Truncating length at $q/100$ -quantiles ($q = 99, 95, 90, 80, 75, 50, 25$). GRU is evaluated at these truncating lengths on each dataset.

B. REARRANGEMENT OF INPUT SEQUENCE

The rearrangement of input sequence makes GRU flexible for handling long input sequences. By rearranging the input sequences, inference can be completed within fixed and limited recurrent steps. This method is used in [15] on RNN,

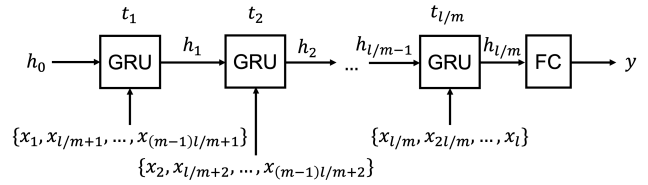


FIGURE 4. GRU inference with rearranged input sequences.

TABLE 3. Truncating length, corresponding dividing part and packing slot for each dataset.

(a) Set-A: recurrent step $L = 5$.

| Dataset | Truncating length | Dividing part | Packing slot |
|--------------|-------------------|---------------|--------------|
| AG_NEWS | 20 | 4 | 64 |
| DBpedia | 15 | 3 | 128 |
| AmazonReview | 50 | 10 | 32 |
| YahooAnswers | 30 | 6 | 64 |
| YelpReview | 123 | 25 | 16 |
| IMDB | 219 | 44 | 8 |

(b) Set-B: recurrent step $L = 11$.

| Dataset | Truncating length | Dividing part | Packing slot |
|--------------|-------------------|---------------|--------------|
| AG_NEWS | 20 | 2 | 128 |
| DBpedia | 15 | 2 | 256 |
| AmazonReview | 50 | 5 | 64 |
| YahooAnswers | 30 | 3 | 128 |
| YelpReview | 123 | 12 | 32 |
| IMDB | 219 | 20 | 16 |

while in [18], a different rearranging scheme is proposed and evaluated on GRU. Here, we employ the same procedure in [18], as illustrated in Fig. 4. Suppose the original length of an input sequence is l , and the dividing part is n , then the length of the new input sequence after rearrangement is $\lceil l/n \rceil$, which significantly reduces the number of recurrent steps required for inference.

We fix the maximum number of recurrent steps L and determine the dividing part n based on the maximum length l_{max} of input sequences in a dataset. This is because in the HE-based GRU, the number of recurrent steps is predetermined by the HE parameters. To test the rearrangement method with different numbers of recurrent steps, we prepare 2 parameter sets with $L = 5$ and 11, respectively. The reason of selecting these 2 values is discussed in IV-D. The evaluation is performed at each truncating length shown in Fig. 3 on each dataset, and the results are shown by the dashed lines in Fig. 3b and Fig. 3c.

From Fig. 3b and Fig. 3c, it can be observed that after rearranging the input sequences, the performance is similar to that without rearranging. Although there is some accuracy degradation, it is acceptable if the truncating length is carefully selected. Comparing these 2 figures, it is also found that the GRU supporting more recurrent steps shows slightly better accuracy performance on most datasets. It indicates that while rearrangement makes it possible to process long input sequences in limited recurrent steps, it is still expected for GRU designs to support more recurrent steps. Additionally, we find that for very long sequences, such as those in the IMDB dataset, the accuracy when truncating at a large length is lower than that at a shorter length. This

suggests the possibility that too many dividing parts may destruct the inside logic of input sequences and degrade accuracy.

From the results in Fig. 3, we select the truncating length according to the following rule. We set the tolerance range of accuracy degradation as 3% compared to the results in Fig. 2b and find the minimum truncating length within this range. If no truncating length exists in this range we select the one with the least accuracy degradation. For each dataset, the truncating length and corresponding dividing part number are summarized in Tab. 3 for $L = 5$ and 11. These 2 parameter sets are referred as Set-A and Set-B, respectively in the following contents.

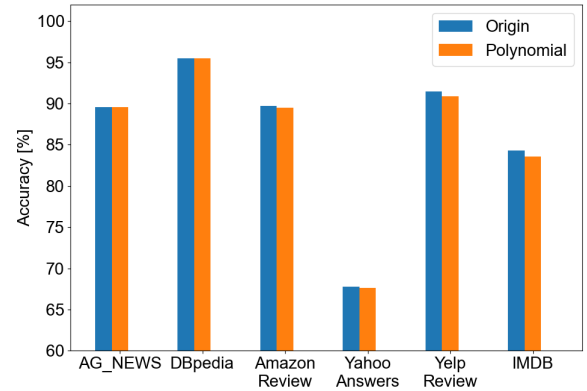
C. POLYNOMIAL APPROXIMATION OF ACTIVATION FUNCTIONS

As only addition and multiplication are supported in HE calculations, all non-linear functions must be replaced by polynomials. Approximating with high-order polynomials improves precision but introduces extra multiplications, which contradicts our design strategy of decreasing N_{re} in Equ. 3. Low-order polynomials can only approximate the non-linear function within a narrow range. When the input falls out of the expected range, significant error is introduced by the polynomial, resulting in a substantial degradation in inference accuracy. Performing normalization before polynomial activation shows the effectiveness in preventing divergence, as demonstrated in [8], [15], and [18]. With performing layer normalization before each activation to narrow the input range to activation functions, even the 2-order polynomial is shown to be sufficient to preserve the accuracy comparable to non-linear functions in [18]. Additionally, the coefficients in each polynomial are trainable, allowing them to be optimized through GRU training to improve accuracy. We evaluate the GRUs with polynomial and original non-linear activation functions, respectively. The other parameters are set as Set-A and Set-B in Tab. 3.

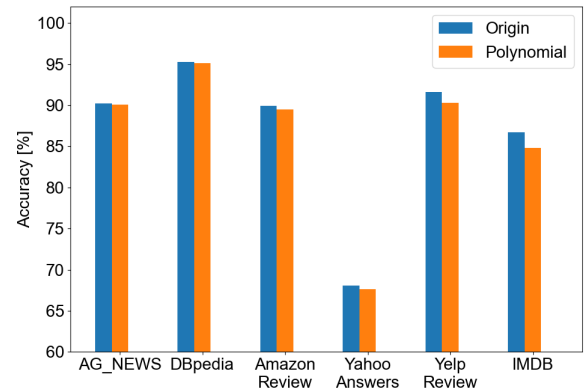
The evaluation results are shown in Fig. 5. Here, despite the more recurrent steps and more complex tasks compared to [18], the 2-order polynomial is still sufficient to preserve accuracy. For both Set-A and Set-B, replacing the non-linear functions with polynomials does not exhibit obvious accuracy degradation. The largest difference is only 2% on IMDB dataset, and on most datasets the differences are below 0.5%, which is almost negligible.

D. HE PARAMETER SELECTION

The selection of HE parameters involves determining the polynomial degree N , the modulus q , the scaling factor Δ and the security level λ . Here we set λ to 128, while N and q are chosen to meet the 128-bit security level. There are 3 objectives in parameter selection: 1) to support more multiplications, 2) to reduce calculation time and 3) to preserve accuracy the same as that on plaintexts. For objective 1), the number of multiplications is constrained by q and Δ . This is because after multiplying two ciphertexts with



(a) Evaluation results on Set-A.



(b) Evaluation results on Set-B.

FIGURE 5. The evaluation results of GRU accuracy with original and polynomial activation function.

scaling factor Δ , the new ciphertext's scaling factor becomes Δ^2 . A rescaling operation is then performed to reduce the scaling factor back to Δ , simultaneously decreasing the modulus q by Δ . Eventually, the ciphertext is not available for multiplication when the modulus is too small to support another rescaling operation. Setting a large q can increase the number of multiplications, but a large N must be selected to meet the λ requirement, leading to slow computation speed. On the other hand, setting a small Δ also increases the number of multiplications but degrades the precision of HE operations. To achieve objective 3), Δ must be selected in a way such that the noise from HE calculations remains small enough compared to the data for computation.

To summarize, the HE parameter selection flow is as follows:

- Select λ to determine the security level (here $\lambda = 128$).
- Select Δ to ensure that HE-based GRU inference gives the same results as those on plaintext.
- Select the maximum q and minimum N under the limitation of security level and computational feasibility.

1) NOISE ANALYSIS

The HE calculations are not totally precise but introduce some error in each operation, which is called noise. Here,

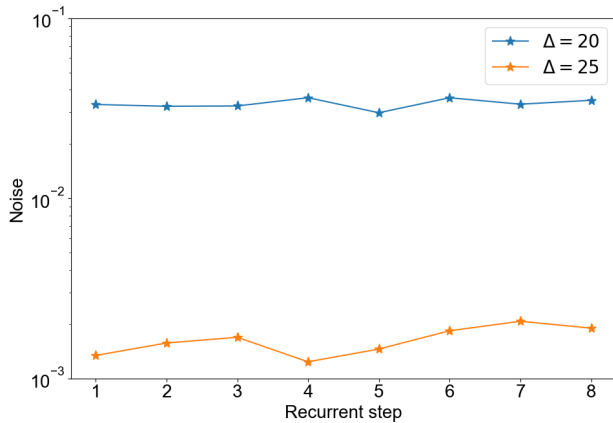


FIGURE 6. Noise from encrypted computations after each recurrent step.

TABLE 4. HE parameters for Set-A and Set-B.

| | $\log N$ | $\log q$ | $\log \Delta$ | Max recurrent step |
|-------|----------|----------|---------------|--------------------|
| Set-A | 15 | 800 | 20 | 5 |
| Set-B | 16 | 1600 | 20 | 11 |

we examine the noise through HE-based GRU inference to decide Δ . We set $\log \Delta$ to 15, 20 and 25 and evaluate the GRU with Set-A. After each recurrent step, we decrypt the ciphertext and compare it with the result from plaintext calculations. The noise of decrypted ciphertexts compared to plaintexts is shown in Fig. 6. The result of $\log \Delta = 15$ is not included because the noise is as significant as data itself, resulting that the calculations are completely incorrect.

Fig. 6 shows that the noise is not accumulating through recurrent steps. While a quantitative analysis is beyond the scope of this paper, a qualitative explanation can be provided. At each recurrent step, input data and GRU parameters are encrypted into fresh ciphertexts. The noise from the last recurrent steps is only contained in the encrypted hidden state. Since the calculations on the hidden state involve fresh ciphertexts, the noise does not accumulate rapidly. As the noise when $\log \Delta = 20$ is always lower than the data by 1 magnitude of degree, it can be ensured that the results from HE calculations are equivalent to those on plaintexts. We also verified the noise when $L = 11$ (Set-B) and the results show that $\log \Delta = 20$ is still sufficient for maintaining precision.

2) PARAMETER SELECTION FLOW

We use the HE-based matrix multiplication algorithm proposed in [18], which requires 2 ciphertext multiplications. By reorganizing the calculations in Equation 1, each recurrent step requires 7 multiplications on 1 ciphertext. We consider setting polynomial degree N to 2^{15} and 2^{16} , which are both typical in HE-based NNs. To meet the $\lambda = 128$ requirement, the modulus q cannot exceed 2^{800} and 2^{1600} , respectively. With setting $\log \Delta = 20$, at most 39/79 multiplications are available, supporting 5/11 recurrent steps. The selected parameters are summarized in Tab. 4.

TABLE 5. Accuracy performance of encrypted GRU and original GRU on each dataset.

| Dataset | Original | Set-A Enc (Degradation) | Set-B Enc (Degradation) |
|--------------------------------|----------|-------------------------|-------------------------|
| AG_NEWS | 91.2 | 89.6 (1.6) | 90.1 (1.1) |
| DBpedia | 96.0 | 95.5 (0.5) | 95.1 (0.9) |
| AmazonReview | 92.0 | 89.5 (2.5) | 89.5 (2.5) |
| YahooAnswers | 68.5 | 67.6 (0.9) | 67.6 (0.9) |
| YelpReview | 94.5 | 90.9 (3.6) | 90.3 (4.2) |
| IMDB | 87.8 | 83.6 (4.2) | 84.8 (3.0) |
| AmazonReview [15] ¹ | 81.9 | 78.4 (3.5) | |
| IMDB [15] | 78.5 | 74.8 (3.7) | |

¹ Only includes the movie and TV show reviews.

V. EVALUATION AND RESULTS

We employ the same encrypting scheme described in [18], where the packing technique is used to perform inference in a SIMD way. Similarly, the translation and embedding of texts are completed by users and encryption is performed on the embedded sequences. The number of packing slots are exponents of 2 and depended by the input and hidden size of GRU, which varies across different datasets (Tab. 2).

We evaluate the HE-based GRU on each dataset with Set-A and Set-B shown in Tab. 3. The experiments are performed on an Apple M2 Max CPU with 32 GB memory. Both data and GRU parameters are encrypted by CKKS scheme. The accuracy of the encrypted GRU, as well as the original GRU, is shown in Tab. 5. After employing the proposed methods for building HE-friendly GRU and performing encrypted calculations, the accuracy degradation is at most 4.2% compared to the original GRU. Notably, for AG_NEWS, DBpedia and YahooAnswers datasets, the degradation is even around 1%. These results proves the effectiveness of our proposed techniques in achieving bootstrapping-free HE-based GRUs while preserving accuracy. The last 2 rows in Tab. 5 shows the results from [15], where an HE-based RNN is implemented. While we train the GRU on each dataset independently, in [15], the results for the IMDB dataset are tested using the GRU trained on the AmazonReview dataset. Our approach demonstrates an 10% accuracy improvement in average, primarily attributable to the more powerful GRU architecture compared to RNN. Given that the architecture of GRU is much more complex than RNN, our results show the availability of building complex models without relying on bootstrapping.

The runtime performance for Set-A and Set-B on each dataset is shown in Tab. 6. The time is not same for all datasets because the complexities of matrix multiplications are different for different packing slot sizes. On Set-A, the runtime is 10-13 minutes for one encrypted GRU inference, while on set-B it is 88-136 minutes. Although on set-B one ciphertext can hold more samples, the throughput is still lower than that on Set-A due to the rapidly increased computing cost. We also compare the runtime results with those reported in [15], where similar parameter sets to Set-A and Set-B are used. Despite the higher complexity of calculations in GRU

TABLE 6. Runtime performance of encrypted GRU on each dataset (unit: sec).

| Dataset | Set-A | | | | Set-B | | | |
|--------------|-------|------------|-----------|-------|-------|------------|-----------|-------|
| | Setup | Encryption | Inference | Total | Setup | Encryption | Inference | Total |
| AG_NEWS | 10 | 44 | 611 | 665 | 58 | 401 | 6050 | 6509 |
| DBpedia | 10 | 45 | 575 | 630 | 57 | 318 | 5303 | 5678 |
| AmazonReview | 11 | 45 | 695 | 751 | 57 | 400 | 6578 | 7035 |
| YahooAnswers | 11 | 47 | 740 | 798 | 57 | 395 | 6013 | 6465 |
| YelpReview | 16 | 45 | 726 | 787 | 57 | 436 | 7541 | 8034 |
| IMDB | 10 | 45 | 819 | 874 | 57 | 437 | 8147 | 8641 |
| [15] | - | - | 2080 | - | - | - | 10527 | - |

compared to RNN in [15], our implementations show a speed improvement of 2.5 to 3.6 times on Set-A and 1.3 to 2.0 times on Set-B. In addition, our memory utilization is significantly lower. In [15], over 100 GB and 200 GB of memory is utilized for performing one inference on Set-A and Set-B, whereas in our experiments, the memory utilization is only 6 GB and 24 GB, respectively.

The runtime can be further improved by parallel computing techniques. While it is challenging to parallelize computations across recurrent steps in a GRU inference due to data dependencies, parallelization within each step is available. For example, computations for r_t and z_t in Equ. 1 can be performed independently and simultaneously at each step. Furthermore, each matrix multiplication comprises several independent ciphertext multiplications and rotations [18]. Given that more than 95% of the runtime in our implementation is attributed to matrix multiplication, parallelizing these computations is expected to significantly enhance computing speed. This parallelization strategy can be particularly effective in optimizing the performance of HE-based GRU inference.

VI. SUMMARY

In this paper, we present a guideline to build bootstrapping-free HE-based GRU for text classification tasks. As the number of multiplications supported in HE-schemes is highly restricted, we introduce several techniques aiming to reduce the required multiplications. First, the text pre-processing techniques are demonstrated. We prove that by removing punctuation marks and function words in texts, the remaining words still possess essential information and the accuracy is almost preserved. Compared to the original texts, the lengths are decreased by 40%-66%, which highly decreases the required recurrent steps. The early truncation is also discussed, which shortens the long sequences to fixed length. The optimized truncating lengths are selected for each dataset to achieve the best accuracy. Second, the rearrangement of input sequences is introduced to enable GRU to process long sequences within fixed recurrent steps. We also show the effectiveness of using 2-order polynomials as activation functions, nearly achieving the minimum polynomial degree. Each recurrent step in GRU only requires 7 multiplications on one ciphertext. At last, we outline an HE parameter selection flow, which ensures the precision of encrypted computations while enabling more multiplications. We analyze the noise

level in ciphertext with different rescaling factors, and decide an optimized one to ensure the encryption does not degrade the computing precision. The proposed HE parameter selection flow is not only available on GRU but any other HE-based applications.

Our evaluation results indicate that the accuracy of our proposed HE-based GRU models is slightly lower than that of the original GRU, with a maximum difference of 4.2%. The runtime for one encrypted inference is only 10-13 minutes on a lightweight parameter set, despite the complex computations in GRU. Our work provides a guideline for building HE-friendly GRU architecture and shows the availability of applying HE to achieve fast and accurate inferences on complex models.

REFERENCES

- [1] J. Domingo-Ferrer, O. Farres, J. Ribes-González, and D. Sánchez, "Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges," *Comput. Commun.*, vols. 140–141, pp. 38–60, May 2019.
- [2] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.* New York, NY, USA: Association for Computing Machinery, May 2009, pp. 169–178.
- [3] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf.* New York, NY, USA: Association for Computing Machinery, Jan. 2012, pp. 309–325.
- [4] J. Fan and F. Vercauteren. (2012). *Somewhat Practical Fully Homomorphic Encryption*. [Online]. Available: <https://eprint.iacr.org/2012/144>
- [5] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology—ASIACRYPT*, T. Takagi and T. Peyrin, Eds., Cham, Switzerland: Springer, 2017, pp. 409–437.
- [6] 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.
- [7] I. Chillotti, M. Joye, and P. Paillier, "Programmable bootstrapping enables efficient homomorphic inference of deep neural networks," in *Cyber Security Cryptography and Machine Learning*, S. Dolev, O. Margalit, B. Pinkas, and A. Schwarzmann, Eds., Cham, Switzerland: Springer, 2021, pp. 1–19.
- [8] 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.
- [9] A. Al Badawi, C. Jin, J. Lin, C. F. Mun, S. J. Jie, B. H. M. Tan, X. Nan, K. M. M. Aung, and V. R. Chandrasekhar, "Towards the AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1330–1343, Jul. 2021.
- [10] 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.

- [11] 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. 39th Int. Conf. Mach. Learn.*, vol. 162, 2022, pp. 12403–12422.
- [12] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 395–412.
- [13] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019.
- [14] J. Jang, Y. Lee, A. Kim, B. Na, D. Yhee, B. Lee, J. H. Cheon, and S. Yoon, "Privacy-preserving deep sequential model with matrix homomorphic encryption," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, May 2022, pp. 377–391.
- [15] R. Podschwadt and D. Takabi, "Non-interactive privacy preserving recurrent neural network prediction with homomorphic encryption," in *Proc. IEEE 14th Int. Conf. Cloud Comput. (CLOUD)*, Sep. 2021, pp. 65–70.
- [16] M. Bakshi and M. Last, "CryptoRNN—privacy-preserving recurrent neural networks using homomorphic encryption," in *Proc. Int. Symp. Cyber Secur. Cryptogr. Mach. Learn.* Cham, Switzerland: Springer, 2020, pp. 245–253.
- [17] B. Feng, Q. Lou, L. Jiang, and G. C. Fox, "CryptoGRU: Low latency privacy-preserving text analysis with GRU," 2020, *arXiv:2010.11796*.
- [18] Z. Wang and M. Ikeda, "High-throughput privacy-preserving GRU network with homomorphic encryption," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2023, pp. 1–9.
- [19] R. Podschwadt, D. Takabi, P. Hu, M. H. Raffei, and Z. Cai, "A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption," *IEEE Access*, vol. 10, pp. 117477–117500, 2022.
- [20] K. Han and D. Ki, "Better bootstrapping for approximate homomorphic encryption," in *Topics in Cryptology—CT-RSA*, S. Jarecki, Ed., Cham, Switzerland: Springer, 2020, pp. 364–390.
- [21] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," in *Advances in Cryptology—EUROCRYPT*, A. Canteaut and F.-X. Standaert, Eds., Cham, Switzerland: Springer, 2021, pp. 587–617.
- [22] Z. Wang and M. Ikeda, "High-throughput and fully-pipelined ciphertext multiplier for homomorphic encryption," *IEICE Electron. Exp.*, vol. 21, no. 6, 2024, Art. no. 20230628.
- [23] Z. Wang and M. Ikeda, "High-throughput key switching accelerator for homomorphic encryption," in *Proc. Int. Conf. IC Design Technol. (ICICDT)*, Sep. 2023, pp. 100–103.
- [24] N. Samardzic, A. Feldmann, A. Krastev, S. Devadas, R. Dreslinski, C. Peikert, and D. Sanchez, "f1: A fast and programmable accelerator for fully homomorphic encryption," in *Proc. MICRO-54: 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021, pp. 238–252.
- [25] T. Shimada and M. Ikeda, "High-speed and energy-efficient crypto-processor for post-quantum cryptography CRYSTALS-kyber," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2022, pp. 12–14.
- [26] N. Samardzic, A. Feldmann, A. Krastev, N. Manohar, N. Genise, S. Devadas, K. Eldefrawy, C. Peikert, and D. Sanchez, "CraterLake: A hardware accelerator for efficient unbounded computation on encrypted data," in *Proc. 49th Annu. Int. Symp. Comput. Archit.*, Jun. 2022, pp. 173–187.
- [27] M. S. Riazi, K. Laine, B. Pelton, and W. Dai, "HEAX: An architecture for computing on encrypted data," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 1295–1309.
- [28] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," 2015, *arXiv:1509.01626*.
- [29] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a web of open data," in *Proc. 6th Int. Semantic Web 2nd Asian Conf. Asian Semantic Web Conf.*, 2007, pp. 722–735.
- [30] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2011, pp. 142–150.



ZEYU WANG received the B.S. degree from the Department of Physics, Zhejiang University, China, in 2019, and the M.E. degree in electronic engineering from The University of Tokyo, Japan, in 2021, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Information Systems. His research interests include the implementation of high-performance homomorphic encryption-based neural networks on both software and hardware.



MAKOTO IKEDA (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in electrical engineering from The University of Tokyo, in 1991, 1993, and 1996, respectively. He joined The University of Tokyo, Tokyo, Japan, as a Research Associate, in 1996. He is currently the Director and a Professor with the Systems Design Laboratory (d.lab), Engineering School, The University of Tokyo. He has been involved in the activities of the VDEC to promote VLSI design educations and research in Japanese academia, since 1996. He has led the AI Chip Design Project for Japanese start-ups, supported by the Ministry of Economy, Trade and Industry of Japan, since 2018. His research interests include hardware security, smart image sensors for 3D range finding, and time-domain circuits, including asynchronous controlling and associate memories. He served as a Distinguished Lecturer for the IEEE Solid-State Circuits Society, from 2013 to 2016, and held numerous positions at international conferences, including the International Solid-State Circuits Conference International Technical Program Committee (TPC) Chair, in 2021, the Program Chair of the Symposium on VLSI Circuits, in 2017, and the TPC Chair of Asian Solid-State Circuits Conference, in 2015. He is a Senior Member of the Institute of Electronics, Information and Communication Engineers, and a member of ACM and the Information Processing Society of Japan.

• • •