Keyword Spotting in the Homomorphic Encrypted Domain Using Deep Complex-Valued CNN

Peijia Zheng[*]
School of Computer Science and Engineering
GuangDong Province Key Laboratory of Information Security Technology
Sun Yat-sen University
Guangzhou, China
zhpj@mail.sysu.edu.cn

Zhiwei Cai[†]
Huicong Zeng[†]
caizhw3@mail2.sysu.edu.cn
zenghc5@mail2.sysu.edu.cn
School of Computer Science and Engineering, Sun Yat-sen University
Guangzhou, China

Jiwu Huang[‡]
Guangdong Key Laboratory of Intelligent Information Processing
Shenzhen Key Laboratory of Media Security
Shenzhen University
Shenzhen, China
jwhuang@szu.edu.cn

ABSTRACT

In this paper, we propose a non-interactive scheme to achieve end-to-end keyword spotting in the homomorphic encrypted domain using deep learning techniques. We carefully designed a complex-valued convolutional neural network (CNN) structure for the encrypted domain keyword spotting to take full advantage of the limited multiplicative depth. At the same depth, the proposed complex-valued CNN can learn more speech representations than the real-valued CNN, thus achieving higher accuracy in keyword spotting. The complex activation function of the complex-valued CNN is non-arithmetic and cannot be supported by homomorphic encryption. To implement the complex activation function in the encrypted domain without interaction, we design methods to approximate complex activation functions with low-degree polynomials while preserving the keyword spotting performance. Our scheme supports single-instruction multiple-data (SIMD), which reduces the total size of ciphertexts and improves computational efficiency. We conducted extensive experiments to investigate our performance with various metrics, such as accuracy, robustness, and F1-score. The experimental results show that our approach significantly outperforms the state-of-the-art solutions on every metric.

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; *Privacy-preserving protocols*; • **Information systems** → *Multimedia information systems.*

KEYWORDS

keyword spotting, complex-valued CNN, homomorphic encryption, deep learning

ACM Reference Format:
Peijia Zheng, Zhiwei Cai, Huicong Zeng, and Jiwu Huang. 2022. Keyword Spotting in the Homomorphic Encrypted Domain Using Deep Complex-Valued CNN. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3503161.3548350

1 INTRODUCTION

With the advent of the big data era, massive amounts of multimedia data are being generated exponentially. As an important part of multimedia data, audio data, such as music and voice, is also growing rapidly in data volume. It is a heavy burden for resource-constrained data owners to store and process an enormous amount of audio data locally. Fortunately, data owners can outsource expensive data storage and processing to public clouds and enjoy convenient services with cloud computing. Typically, public clouds are managed by third parties. Outsourcing data to public clouds without protection will raise security concerns for users on their uploaded data. It is therefore desirable to develop privacy-preserving processing in cloud computing. The methods based on homomorphic encryption, e.g., signal processing in the encrypted domain, are promising approaches among existing privacy-preserving processing techniques.

Homomorphic encryption (HE) plays an important role in privacy-preserving processing techniques. Existing HEs can be roughly divided into two types: partially homomorphic encryption [37, 39] and fully homomorphic encryption [22]. The algorithm executors can manipulate the HE ciphertexts so that the plaintexts are computed according to the algorithm. Suppose that users upload the HE ciphertexts of their data, the cloud server can then perform privacy-preserving applications on these HE ciphertexts without knowing users' private data.

There are already many privacy-preserving applications for image data based on HE, such as privacy-preserving image classification [11, 23, 24], privacy-preserving human action recognition [32], etc. There are a few works on privacy-preserving processing, e.g., privacy-preserving speech recognition [48]. Nevertheless, this work assumes that the features are extracted in the plaintext domain, and

[*]Also with Zhengzhou Xinda Institute of Advanced Technology, Zhengzhou, China.
[†]Both authors contributed equally to this research.
[‡]Also with Shenzhen Institute of Artificial Intelligence and Robotics for Society, China.
- This work was supported in part by the National Key Research and Development Project (2020YFB1805404), NSFC (U19B2022), Guangdong Basic and Applied Basic Research Foundation (2019A1515010746, 2022A1515011897), Science and Technology Projects in Guangzhou (202102080354), Open Foundation of Henan Key Laboratory of Cyberspace Situation Awareness (HNTS2022023).
- *Corresponding Authors: Jiwu Huang.*

the speech recognition algorithm is performed on the encrypted features. It is desirable to develop non-interactive privacy-preserving approaches for important speech processing techniques (e.g., keyword spotting) over HE ciphertexts of speech data. As an important application in speech processing, keyword spotting aims at detecting a pre-defined keyword or a set of keywords in a continuous stream of audio [19]. Keyword spotting has plenty of applications such as speech data mining, audio indexing, phone call routing, activating voice assistants, etc [51]. The recent development of deep learning has led to greater advances in end-to-end keyword spotting. Considering its importance in speech processing, it is meaningful to perform keyword spotting in the HE domain. However, to the best of our knowledge, there is no report on end-to-end keyword spotting in the HE domain.

It is a great challenge to enable effective end-to-end keyword spotting in the HE domain. Due to the limitation of the current HE scheme, the maximum allowable multiplicative depth in the HE domain is limited. Although bootstrapping can be used to refresh ciphertexts and increase the multiplicative depth, it takes too much time and has poor practicality. Therefore, we focus on the employment of leveled homomorphic encryption (LHE) and do not consider the use of bootstrapping in the following. The effectiveness of traditional keyword spotting schemes generally depends on speech feature extractors and machine learning algorithms. Traditional speech feature extractors (e.g., MFCC) and machine learning algorithms (e.g., SVM) require a large multiplicative depth, which exceeds the maximum multiplicative depth of the current HE scheme. Therefore it is difficult to implement traditional keyword spotting schemes non-interactively in the encrypted domain. It is more promising to develop a secure keyword spotting scheme using deep learning techniques.

The deep networks of existing end-to-end keyword spotting schemes are too deep to be directly migrated to the encryption domain of LHE. If we shorten these deep networks before migrating them to the HE domain, it may lead to a much lower performance of keyword spotting. Although there are some works on implementing deep neural networks for secure classification in the HE domain, they are optimized for image datasets. Applying these schemes to the keyword spotting task does not yield good results. In addition, all the existing deep networks in the encrypted domain use real-valued deep networks. At the same depth, the complex-valued deep network can process twice as much input data as the real-valued deep network, which means that the complex-valued deep network can learn more representational information from the data. Therefore, the use of complex-valued deep networks in keyword spotting has the potential to improve recognition performance. To our best knowledge, there are no reports on complex-valued deep networks in the encrypted domain. It is necessary and meaningful to design a complex-valued CNN structure for the encrypted domain keyword spotting under the constraint of the current HE schemes.

In this paper, we attempt to solve this problem by proposing an end-to-end keyword spotting scheme in the encrypted domain using deep learning techniques. We propose Crypto$\mathbb{C}$Net, a solution to the complex-valued convolutional neural network (CNN) algorithms on encrypted data. Compared with the commonly used real-valued CNNs, the complex-valued CNNs have better feature capturing capability with the same depth. We employ the CKKS scheme as the cryptographic primitive, which is a famous approximate HE. The CKKS scheme supports approximate arithmetics over complex numbers that are suitable for machine learning. We present the implementation of the building blocks in the HE domain, including the complex-valued convolutional layer, the pooling layer, and the fully connected layer. Then, we propose a method for approximating complex activation functions using low-degree polynomials while preserving accuracy. With the approximate polynomial activation function, we can perform complex-valued CNNs in the HE domain without any interaction. For the keyword spotting task, we design the network structure of the complex-valued CNN to evaluate in the encrypted domain and maintain high recognition performance. The proposed Crypto$\mathbb{C}$Net also employs the single-instruction multiple-data (SIMD) technique, which allows multiple audio samples to be processed simultaneously.

We summarize our contributions as follows.

- We propose the implementation of the complex-valued convolutional layer, the pooling layer, and the fully connected layer in the HE domain, using the SIMD technique that allows the simultaneous batching of multiple samples.
- We investigate several methods for approximating the commonly used complex activation functions with low-degree polynomials in complex-valued CNNs, which allows activation functions to be implemented non-interactively in the HE domain.
- We design a network structure for complex-valued CNNs for keyword spotting in the HE domain. To our best knowledge, there is no previous report on complex-valued deep networks in the HE domain.
- We conducted extensive comparison experiments on the Speech Commands dataset. The experimental results show that our scheme significantly outperforms the state-of-the-art works in various metrics.

The rest of this article is organized as follows. In section 2, we review related works about keyword spotting in the plaintext domain and neural networks in the encrypted domain. Section 3 introduces the preliminaries related to this paper. Section 4 presents building blocks of our secure complex-valued CNN. In section 5, we give a security analysis of the proposed scheme and cryptographic parameters analysis. Section 6 provides experimental results. In section 7, we summarize the paper and discuss the future work direction.

2 RELATED WORKS

2.1 Keyword Spotting in the Plaintext Domain

Various traditional methods are proposed for keyword spotting. Senthildevi *et al.* proposed a Tamil keyword recognition system using MFCCs and dynamic time warping algorithms [1]. The works in [21, 30, 40] described some hybrid methods of SVM, hidden Markov model, and Gaussian mixed model. Recently, motivated by the impressive success of deep learning, researchers have begun to propose keyword spotting schemes based on deep networks. In [41], Tang *et al.* used MFCC to extract features beforehand and explored the use of deep residual networks with dilated convolutions. By using a short-time Fourier transform module to form the front-end and a simple two-dimensional CNN to form the back-end, Won *et al.* [45] proposed a keyword spotting algorithm. In [19], Coucke *et al.* introduced end-to-end stateless modeling for keyword spotting, based on dilated convolutions coupled with residual connections and gating. In [36], Mittermaier *et al.* employed depth-wise separable convolutions (DSConv) [29] to constitute an end-to-end keyword spotting architecture. Choi *et al.* proposed [18] a ResNet-inspired architecture convolved along the time dimension and treat the frequency dimension as channels. In [33], Lin *et al.* exploited a pre-trained speech embedding model trained to extract useful features for keyword spotting models. The scheme in [35] uses synthetic speech for data enhancement to strengthen the model. Raju *et al.* [35] presented a similar multi-target training setup for their keyword spotter. However, the above methods are designed for unencrypted audio data. The multiplicative depth of their algorithms is too high to be migrated to the HE domain. We need to propose a practically usable keyword spotting scheme for the HE domain.

2.2 Deep Learning in the Encrypted Domain

Xie *et al.* [46] discussed the theoretical aspects of implementing neural networks using polynomial approximations in the encrypted domain. Building on this work, Dowlin *et al.* [23] proposed CryptoNets, a neural network classifier on encrypted data. CryptoNets use the square activation layer as the activation function layer, replacing the max pool layer with the average pool layer. Chabanne *et al.* [11] combined the idea of CryptoNets with batch normalization to improve the accuracy of CryptoNets. Hesamifard *et al.* [24] compared the performance of activation functions approximated by different low-order polynomials and found that the derivative approximation performed best. Ishiyama *et al.* [26] compared the performance of Swish and ReLU activation functions by polynomial approximation in different intervals. In [5, 28, 38], secure schemes based on secure multi-party computing techniques are proposed. But these methods require a large amount of data transmission between the client and the server, which brings a high communication cost. Badawi *et al.* [3] proposed a very efficient GPU implementation to achieve high performance. In [10, 12, 27], efficient evaluation of neural networks are proposed by optimizing coding and matrix multiplication. However, all the works above deal with the MNIST image dataset other than the audio data. Prior arts offer no solutions to enable secure speech spotting directly over HE ciphertexts of audio data. We propose a HE-friendly complex-valued CNN for speech spotting in the HE domain to address these issues.

3 BACKGROUND

3.1 Homomorphic Encryption

We can group existing HEs into two groups: partial homomorphic encryption (PHE) and fully homomorphic encryption (FHE). PHE allows homomorphic addition or multiplication. For example, Paillier encryption [37] is the most famous additive HE that are widely used in various privacy-preserving schemes [4, 49, 50]. The unsupported non-linear operations of the Paillier encryption are implemented by multi-party computation (MPC) [2, 7, 47], which is typically interactive with the party holding the private key during computation.

FHE supports an arbitrary number of homomorphic additions and multiplications, with a costly bootstrapping procedure to refresh ciphertexts to increase the multiplicative depth. The bootstrapping procedure is so time-consuming that it is not yet practical. Therefore, we focus on LHE that supports homomorphic evaluation of polynomials with a bounded degree, such as YASHE [6], BGV [9], BFV [8, 20], TFHE [16] and CKKS [15]. The CKKS scheme is a word-wise approximate FHE scheme that supports the evaluation of encrypted real or complex data. We can construct the CKKS scheme based on ring learning with errors problem (RLWE) [34]. For a power-of-two integer N, we use $\mathcal{R} = \mathbb{Z}[X]/(X^N+1)$ to denote the cyclotomic polynomial ring of dimension N, where $\mathbb{Z}[X]$ is the polynomial ring with integer coefficients. For a positive integer l, we use $\mathcal{R}_l = \mathcal{R}/2^l\mathcal{R}$ to denote the residue polynomial ring of $\mathcal{R}$. The plaintext and the ciphertext spaces are defined in $\mathcal{R}$ and $\mathcal{R}_l$, respectively. For a positive integer $k \le N/2$, the CKKS scheme provides a technique to pack k complex numbers in a single polynomial using a variant of the complex canonical embedding map $\phi : \mathbb{C}^k \to \mathcal{R}$. With this Single Instruction-Multiple Data (SIMD) technique, the CKKS scheme allows multiple plaintexts to be processed simultaneously in the HE domain in a batch fashion.

Suppose $\mathbf{x} = (x_0, x_1, \ldots, x_{k-1})$ and $\mathbf{y} = (y_0, y_1, \ldots, y_{k-1})$ are two complex-valued vectors, where $k \le N/2$. For convenience, we omit the encoding symbol ϕ and use $[\![\cdot]\!]$ to represent the encryption operator of the plaintext vector, e.g., $[\![\mathbf{x}]\!]$ means a ciphertext of the plaintext vector $\mathbf{x}$. We summarize some important SIMD-type homomorphic operations of CKKS as follows:

$$\begin{aligned}
[\![\mathbf{x}]\!] \oplus [\![\mathbf{y}]\!] &= [\![(x_0 + y_0, x_1 + y_1, \ldots, x_{k-1} + y_{k-1})]\!] \\
[\![\mathbf{x}]\!] \otimes [\![\mathbf{y}]\!] &= [\![(x_0 \times y_0, x_1 \times y_1, \ldots, x_{k-1} \times y_{k-1})]\!] \\
[\![\mathbf{x}]\!] \odot z &= [\![(x_0 \times z, x_1 \times z, \ldots, x_{k-1} \times z)]\!],\ z \in \mathbb{C}
\end{aligned}$$

where $\otimes$ and $\odot$ denote the scalar multiplication and non-scalar multiplication, respectively. Non-scalar multiplication costs significantly more runtime than scalar multiplication. The readers may refer to [13–15] for more technical details and security analysis of the CKKS scheme.

3.2 Complex-Valued Convolutional Neural Networks

The complex-valued CNN consists of complex-valued neurons organized in multiple layers [42]. Each complex-valued neuron maps the input to the output by executing a function. Input and output numbers are represented in complex-valued forms. The structure of the function is determined by the network model layer to which it belongs. There is at least one hidden layer between the input and output layers composed of various types of complex-valued components. Complex-valued CNN commonly includes complex-valued convolutional layers, complex-valued activation functions, and complex-valued pooling layers.

We introduce the representation of complex numbers in the network as the start. A complex number $z = a + \imath b$ has a real part a and an imaginary part b, which are real numbers. Following [17, 25], we represent the real part a and the imaginary part b as logically distinct real values when training the network. After training, we can learn the real and imaginary part weights, which are then combined into the complex-valued weights. During the prediction stage, the complex-valued weights are used to instantiate complex-valued components of the complex-valued CNN. We provide a brief introduction on these components in the following.

3.2.1 Complex-Valued Convolutional Layer. A complex-valued convolutional layer is a set of complex-valued filters operating on the complex-valued input. Each complex-valued filter is represented as a complex-valued matrix. We get the convolutional result of one point by computing the dot product of the filter's complex-valued weights and the two-dimensional input values in the neighbor of that point.

We denote $\Re(\cdot)$ and$\Im(\cdot)$ as the real and imaginary parts of a complex number, respectively, and denote $*$ as convolution operator. We convolve a two-dimensional complex-valued input matrix $\mathbf{M} = \mathbf{X} + \imath\mathbf{Y}$ by a 2D complex-valued filter $\mathbf{K} = \mathbf{A} + \imath\mathbf{B}$, where $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{A}$, and $\mathbf{B}$ are all real-valued matrices. Since the convolution operation is distributive, convolving the complex-valued input $\mathbf{M}$ by the filter $\mathbf{K}$ can be decomposed into convolutions of real-valued matrices, i.e.,

$$\mathbf{M} * \mathbf{K} = (\mathbf{X} * \mathbf{A} - \mathbf{Y} * \mathbf{B}) + \imath(\mathbf{X} * \mathbf{B} + \mathbf{Y} * \mathbf{A}) \tag{1}$$

Thus, we get the equivalent representation of a complex-valued convolution as the traditional real-valued convolution.

3.2.2 Complex-Valued Pooling Layer. The complex-valued pooling layer is usually used immediately after the complex-valued activation layer in complex-valued CNN. The complex-valued pooling layer is for sub-sampling from the data to reduce the size of parameters of the neural network. This layer divides the data into non-overlapping subareas and performs a function to obtain a value on each sub-area. One of the most common pooling methods are average pooling, which outputs the average value of the subarea.

3.2.3 Complex-Valued Activation Layer. The complex-valued activation layer is a non-linear complex function, which can enhance the expression ability of the model. Each complex activation function takes a single complex number as input and performs some arithmetic operation to obtain a complex result. Many complex activation functions have been proposed based on the ReLU. Trabelsi *et al.* [42] proposed one complex activation functions $\mathbb{C}$ReLU. Given a complex number z, $\mathbb{C}\mathrm{ReLU}(z)$ will separate z's real and imaginary parts and then perform ReLU functions on them, i.e.,

$$\mathbb{C}\mathrm{ReLU}(z) = \mathrm{ReLU}(\Re(z)) + \imath\mathrm{ReLU}(\Im(z)),\ z \in \mathbb{C}. \tag{2}$$

3.2.4 Complex-Valued Fully Connected Layer. In the fully connected layer, each neuron is connected to all neurons in the previous layer. Each connection is associated with a complex-valued weight. The output of each neuron in this layer is the dot product of two vectors, which are the outputs of neurons in the previous layer and the attached weights of connections, respectively. Similarly, we can get the equivalent representation of vector multiplication between a complex number and real number. We assume that there are a one-dimensional complex input vector $\mathbf{m} = \mathbf{x} + \imath\mathbf{y}$ and a one-dimensional complex-valued weights $\mathbf{k} = \mathbf{a} + \imath\mathbf{b}$, where $\mathbf{x}$, $\mathbf{y}$, $\mathbf{a}$, and $\mathbf{b}$ are all real-valued vectors. We calculate the inner product of $\mathbf{m}$ and $\mathbf{k}$ as

$$\mathbf{m} \star \mathbf{k} = (\mathbf{a} \star \mathbf{x} - \mathbf{b} \star \mathbf{y}) + \imath(\mathbf{b} \star \mathbf{x} + \mathbf{a} \star \mathbf{y}) \tag{3}$$

Algorithm 1 Complex-valued Convolution in the HE domain

```
1:  Input: $\mathbf{C}_{11}, \dots, \mathbf{C}_{1N_{in}}, \mathbf{C}_{21}, \dots, \mathbf{C}_{N_{out}N_{in}} \in \mathbb{C}^k$ and matrix
    $\mathbf{Z}$ of $N_{in}$ ciphertext vectors.
2:  Output: matrix $\tilde{\mathbf{Z}}$ of $N_{out}$ ciphertext vectors..
3:  for $j = 1, 2, \cdots, N_{out}$ do
4:      for $l = 1, 2, \cdots, n - k + 1$ do
5:          for $o = 1, 2, \cdots, N_{in}$ do
6:              for $m = 1, 2, \cdots, k$ do
7:                  $\tilde{\mathbf{Z}}_{jl} = \mathbf{C}_{jom} \odot \mathbf{Z}_{o(l+m-1)} \oplus \tilde{\mathbf{Z}}_{jl}$
8:              end for
9:          end for
10:     end for
11: end for
12: return $\tilde{\mathbf{Z}}$
```

where $\star$ denote the vector multiplication operator. The readers may refer to [42] for complete technical details of complex-valued CNN.

4 PROPOSED SCHEME

We present our building blocks in the HE domain, which are employed to construct our secure complex-valued CNN. Each building blocks are implemented with the SIMD technique, so that multiple plaintexts can be processed simultaneously in the HE domain.

4.1 Complex-Valued Convolutional Layer

We first explain the complex-valued convolution of a single input with a single channel. We use $\mathbf{z}_i$ to denote the vector of complex numbers $(z_{i1}, z_{i2}, \dots, z_{ik})$. The input is a vector of ciphertexts $\mathfrak{Z} = ([\![\mathbf{z}_1]\!], [\![\mathbf{z}_2]\!], \dots, [\![\mathbf{z}_n]\!])$. We assume that the weights of the complex-valued kernel has been obtained after the training stage. Since the real and imaginary weights are learned separately in the training, we will have the real part $\mathbf{r} = (r_1, r_2, \dots, r_k) \in \mathbb{R}^k$ and the imaginary part $\mathbf{i} = (i_1, i_2, \dots, i_k) \in \mathbb{R}^k$, where k is less than n. We then combine $\mathbf{r}$ and $\mathbf{i}$ to obtain the weights of the complex-valued convolutional kernel as $\mathbf{C} = \mathbf{r} + \imath\mathbf{i}$. As the complex-valued kernel slides through the input vector, we compute the dot product of the complex-valued kernel with the sub-vectors covered by the kernel on the input vector. The result of the single-channel complex-valued convolution remains a vector of ciphertexts.

We consider the case of multiple output convolution with multiple input channels. We assume that the numbers of input channels and filters are N_{in} and N_{out}, respectively. Each filter consists of N_{in} complex-valued kernels. Every complex-valued kernel will convolve one input channel to obtain a vector of ciphertexts with the single-channel convolution method. Thus, N_{in} input channels and N_{in} complex-valued kernels will generate N_{in} vectors. We then perform summation on the resultant N_{in} vectors over the channels to get one output vector of ciphertexts. After repeating this process with the remaining filters, we can obtain N_{out} vectors of ciphertexts as the output. Therefore, the input is a matrix $\mathbf{Z}$ of N_{in} ciphertext vectors, i.e., $\mathbf{Z} = (\mathfrak{z}_1^T, \mathfrak{z}_2^T, \dots, \mathfrak{z}_{N_{in}}^T)^T$, where ^{T} denotes the transpose of a matrix or vector. The output is a matrix $\tilde{\mathbf{Z}}$ of N_{out} ciphertext vectors. The explicit steps of multiple output complex-valued convolution in the HE domain are depicted in Algorithm 1.

4.2 Complex Activation Function

The complex activation function plays an important role in the complex-valued CNN in the HE domain. An appropriate complex activation function and its accurate implementation in the HE domain will greatly benefit the speech spotting performance in the HE domain. In the following, we will introduce our complex activation function and its approximation method, and then describe the implementation of the complex activation function in the HE domain.

4.2.1 Polynomial Approximation in the Complex Field. The real-valued activation function ReLU is widely used in deep networks. However, ReLU is a non-linear function that cannot be realized in the HE domain. To implement ReLU in the HE domain, some approximation methods have been proposed, such as the least square method [11] and the derivative approximation method [24]. The least square method first performs data sampling on the activation function and then uses the least square method to obtain the approximate polynomial. The derivative approximation method first computes the approximation of the derivative of the activation function and then obtains the approximation of the activation function by integration. Both these two methods use polynomials to approximate activation functions, so that the approximate polynomial activation functions can be implemented in the HE domain with the homomorphic properties. For convenience, we denote the polynomial approximation function of the ReLu function as PReLU. After performing the least square method or the derivative approximation method, we can obtain the PReLU function as

$$\text{PReLU}(x) = a_0 + a_1 x + \ldots + a_n x^n, \tag{4}$$

where n is the degree of PReLU(x), and $x, a_1, a_2, \ldots, a_n \in \mathbb{R}$. The degree of PReLU cannot be very high due to the limited multiplicative depth of the HE.

We focus on the polynomial approximation method of the complex activation function using the approximation methods of the real-valued activation function. We propose two methods for approximating complex activation functions.

(1) We consider the complex activation function as a combination of two real-valued activation functions. We separately approximate the real and imaginary parts to obtain two polynomial functions, and then combine them to obtain the approximation function of the complex activation function. For example, when we adopt $\mathbb{C}$ReLU as the activation function, the approximation function can be obtained as

$$\text{P}\mathbb{C}\text{ReLU}(z) = \text{PReLU}(\Re(z)) + \imath\text{PReLU}(\Im(z)),\ z \in \mathbb{C}. \tag{5}$$

(2) The input of the polynomial approximation function PReLU of the ReLU function is a real number. Although the domain of PReLU is real filed, we can adapt PReLU for complex numbers by extending its domain to the complex field, i.e.,

$$\text{PEReLU}(z) = a_0 + \imath a_0 + a_1 z + \ldots + a_n z^n,\ z \in \mathbb{C}. \tag{6}$$

4.2.2 Implementing Complex Activation Function in the HE Domain. In the encryption domain of RLWE-based HE, arithmetic operations will result in increased noise. Homomorphic multiplication introduces more noise than homomorphic addition. Typically, the maximum allowable multiplicative depth of the current HE is limited. Compared with high-degree approximated polynomials, low-degree polynomials of activation function allows for implementing a deep network in the HE domain with more depth. Thus, following [23], we adopt third-degree polynomials as the approximated activation functions in our implementation. Let us denote the approximate activation function by $f_a(z) = a_3 z^3 + a_2 z^2 + a_1 z + a_0 + \imath a_0$. The SIMD-type evaluation of the approximate activation function is given as

$$a_3 \odot [\![\mathbf{z}]\!]^3 \oplus a_2 \odot [\![\mathbf{z}]\!]^2 \oplus a_1 \odot [\![\mathbf{z}]\!] \oplus a_0, \tag{7}$$

whose multiplicative depth is three. We can firstly calculate $[\![a_3 x]\!]$ and $[\![x^2]\!]$, respectively. Then, we calculate $[\![a_3 x^3]\!]$ with the ciphertext multiplication. With this strategy, we reduce the required multiplicative depth to two.

We provide some example results of P$\mathbb{C}$ReLU and PEReLU in Table 1. We have also tested these approximated activation functions in Crypto$\mathbb{C}$Net. T he results are given in Table 3.

Table 1: Polynomial approximation result for $z = x + \imath y$.

Activation function	Approximation method	Polynomial approximated
P$\mathbb{C}$ReLU	Least square method	$2.025e^{-6}x^3 + 0.156x^2 + 0.499x + 0.281 + \imath(2.025e^{-6}y^3 + 0.156y^2 + 0.499y + 0.281)$
	Derivative approximation	$6.250e^{-5}x^3 + 0.375x^2 + 4.999x + 0.125 + \imath(6.250e^{-5}y^3 + 0.375y^2 + 4.999y + 0.125)$
PEReLU	Least square method	$6.835e^{-6}z^3 + 0.234z^2 + 0.499z + 0.187 + 0.187\imath$
	Derivative approximation	$1.851e^{-4}z^3 + 0.250z^2 + 0.499z + 0.180 + 0.180\imath$

4.3 Complex-Valued Pooling Layer

In the complex-valued pooling layer, we perform down-sampling of the vector of ciphertexts $\mathfrak{Z} = ([\![\mathbf{z}_1]\!], [\![\mathbf{z}_2]\!], \ldots, [\![\mathbf{z}_n]\!])$. Existing FHE schemes currently do not support the comparison operation, so we only use average-pooling in the pooling layer in the HE domain. The pooling window of length k will slide k steps on the ciphertext vector at each time. We calculate the encrypted mean value of k ciphertext data covered by the window at every last sliding position. The output is a ciphertext vector of $\lfloor \frac{n}{k} \rfloor$ encrypted data, where $\lfloor \cdot \rfloor$ denote rounding down operation. We depict the explicit steps of this SIMD-manner pooling process in Algorithm 2.

4.4 Complex-Valued Fully Connected Layer

In complex-valued fully connected (FC) layer, each complex-valued neuron connects to all neurons in the previous layer with the attached weight. The convolutional neurons in the former layer should first be channel-wise arranged in row-major order and then connected. The input is a ciphertext vector $\mathfrak{z} = ([\![\mathbf{z}_1]\!], [\![\mathbf{z}_2]\!], \ldots, [\![\mathbf{z}_n]\!])$ from the previous complex-valued neurons. The separately trained

Algorithm 2 Pooling in the Encrypted domain

```
1: Input: $[\![z_1]\!], [\![z_2]\!], \cdots, [\![z_n]\!]$, and $k \in \mathbb{R}$.
2: Output: $[\![\tilde{z}_1]\!], [\![\tilde{z}_2]\!], \cdots, [\![\tilde{z}_{\lfloor \frac{n}{k} \rfloor}]\!]$.
3: for $j = 1, 2, \cdots, \lfloor \frac{n}{k} \rfloor$ do
4:     for $l = (j-1) \times k, (j-1) \times k + 1, \cdots, j \times k - 1$ do
5:         $[\![\tilde{z}_j]\!] = \frac{1}{k} \odot [\![z_l]\!] \oplus [\![\tilde{z}_j]\!]$
6:     end for
7: end for
8: return $[\![\tilde{z}_1]\!], [\![\tilde{z}_2]\!], \cdots, [\![\tilde{z}_{\lfloor \frac{n}{k} \rfloor}]\!]$
```

real part $\mathbf{R} = (r_1, r_2, \ldots, r_k)$ and imaginary part $\mathbf{I} = (i_1, i_2, \ldots, i_k)$ are combined to form the complex-valued weight $\mathbf{W} = \mathbf{R} + \imath\mathbf{I}$. A subroutine is to compute the dot product of the subsidiary complex-valued weight vector of a neuron with the ciphertext input vector. We will obtain the output vector of ciphertexts by repeating this subroutine for each neuron in that layer in turn.

4.5 Network Architecture and Training

The recent work CryptoNets use two different network structures in training and prediction. Specifically, they use a nine-layer network for training and a simpler five-layer network for prediction. Different from CryptoNets, we use the same network structure in training and prediction. Therefore, our method ensures that the network can maintain the same performance in prediction as in training. The architecture, parameters, and output size of our Crypto$\mathbb{C}$Net are shown in Table 2, where the input to the network is 8000 pieces of data.

During network training, each layer is required to simulate complex-valued arithmetic operations using both real and imaginary parts. Then, we can convert the training of Crypto$\mathbb{C}$Net into the training of real-valued CNN as [43]. Dropout layers are used when we train the model to avoid over-fitting, and we remove them in the prediction step. After network training, the complex-valued parameters of Crypto$\mathbb{C}$Net are obtained by combining the real and imaginary parameters of the corresponding layer in the training networks.

Table 2: Our network architecture

Layer	Parameters	Output size
1st Conv layer	32 filters of size 13	32×7988
Activation layer	-	32×7988
Average pooling layer	Pool size 3	32×2662
2nd Conv layer	32 filters of size 11	32×2652
Activation layer	-	32×2652
Average pooling layer	Pool size 3	32×884
3rd Conv layer	64 filters of size 9	64×876
Activation layer	-	64×876
Average pooling layer	Pool size 3	64×292
4th Conv layer	128 filters of size 7	128×286
Activation layer	-	128×286
Average pooling layer	Pool size 3	128×95
5th Conv layer	128 filters of size 5	128×91
Activation layer	-	128×91
Average pooling layer	Pool size 3	128×30
6th Conv layer	192 filters of size 3	192×28
Activation layer	-	192×28
Average pooling layer	Pool size 3	192×9
FC layer	256 units	1×256
Activation layer	-	1×256
output layer	35 units	1×35

5 DISCUSSIONS

5.1 Security Analysis

We use the following theorem to analyze the security of our scheme under the Chosen Plaintext Attack (IND-CPA).

Theorem 1. *Our scheme satisfies the IND-CPA security.*

Proof. In our scheme, the audio data is encrypted by the CKKS scheme before the cloud server processes it. During the whole execution of the deep network, the cloud server only operates on the ciphertext data using homomorphism and cannot perform decryption (no decryption key). Therefore, the security of our scheme is equivalent to the security of the employed HE scheme. Since the CKKS scheme is proven to be secure under the CPA, our scheme satisfies the IND-CPA security. □

The cloud server may try to use brute force to obtain the original plaintext audio. Let us consider n seconds of audio at a sampling frequency of 8000Hz. There are $n \times 8000$ ciphertexts, where each ciphertext encrypts m audio signals. We assume that we represent each audio data with a 64-bit digit, there are $2^{n \times 8000 \times m \times 64}$ different possibilities for restoring the original plaintext audio. When $n = 1, m = 1024$, the cloud server needs $2^{1 \times 8000 \times 1024 \times 64}$ operations to restore the original plaintext audio, which is very difficult for existing computers. Thus, the cloud server cannot break the proposed scheme in the encrypted domains even with a brute force attack.

5.2 Cryptographic Parameters Analysis

We denote the bit-length of a fresh ciphertext modulus by L. After constant multiplication or ciphertext multiplication, we need to perform Rescale operation to ensure that the decryption is correct. Let us denote the bit-lengths of constant multiplication and ciphertext multiplication by p_c and p, respectively. After constant multiplication, the ciphertext modulus will be reduced by p_c bits. While after the ciphertext multiplication operation, the ciphertext modulus will be reduced by p bits. We will perform these operations at every layer of our complex-valued CNN. The convolutional layer, the pooling layer, and the fully connected layer each involve a constant multiplication. In Eq. (7), the complex-valued activation layer needs to compute $[\![a_3 x^3]\!]$. We can first calculate $[\![a_3 x]\!]$ and $[\![x^2]\!]$ separately and then calculate $[\![a_3 x^3]\!]$ with a ciphertext multiplication. With this approach, the multiplicative depth is only two. Thus, in the whole Crypto$\mathbb{C}$Net, the ciphertext modulus will reduced by

$$L_1 = 6 \times (p_c + p_c + p_c + p) + 2p_c + p + p_c = 21p_c + 7p. \quad (8)$$

For example, we can set $L = 1200$, $p = 30$ and $p_c = 30$. Then, we leave $1200 - (21 + 7) \times 30 = 360$ bits for ciphertext modulus to get

Table 3: The accuracy of four activation functions for keyword spotting.

Activation function	Approximation method	Accuracy(%)
Pℂ ReLU	Least square method	74.1
	Derivative approximation	73.9
PEReLU	Least square method	**74.4**
	Derivative approximation	74.1

the correct decryption result, which is enough for high precision computing for speed processing.

6 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we conduct experiments to evaluate the effectiveness of our CryptoℂNet for keyword spotting in the HE domain. All experimental results are generated on a 64-bit Ubuntu18.04.5LTS PC with Intel Xeon Gold 6145 CPU @2.00GHZ and 125G RAM. We have implemented the proposed scheme in C++ based on the HEAAN [31] library, which implements the CKKS cryptosystem. We choose the bit-length of the initial ciphertext module as $L = 1200$, the dimension of the cyclotomic ring $\mathcal{R}$ as $N = 2^{11}$. There are 2^{10} plaintext slots in a ciphertext. We also set $p = 30$ and $p_c = 30$ to assure high-precision data processing as analyzed in Section 5.2.

6.1 Dataset

The Speech Commands dataset [44] is built for detecting the spoken word in keyword spotting. Each keyword in the dataset consists of one-second speech data from various speakers. A total of 105,829 speech data are used for training and prediction, and each data is labeled with one of thirty-five keywords. We resample the speech data at a frequency of 8000Hz and obtain 8000 real numbers as the size of the model input.

6.2 Effects of different Activation Functions

In Section 4.2, we have presented two methods to approximate complex activation functions. We have also provided four specific polynomial activation functions in Table 1. We now compare the performance of these four activation functions in the proposed network structure for keyword spotting. The comparison results are shown in Table 3. With the same polynomial approximation methods (i.e., the least square method and the derivative approximation method), the activation function PERELU is always better than PℂRELU. For example, when adopting the same least square method, PℂRELU achieves the accuracy of 74.1%, while PERELU achieves the accuracy of 74.4%, which is the best among the four compared activation functions. Therefore, we use PERELU with least square method as the activation function for our CryptoℂNet in the following discussions.

6.3 Running Time

The evaluation of our complex-valued CNN in the HE domain in the Speech Commands dataset costs 16796.5657 seconds. Since our scheme supports the batch processing manner with the SIMD technique, we can decrease the amortization time for each audio signal. For example, in our experiment, we choose $N = 2^{11}$ for the CKKS scheme and thus allow the batch size of 1024. Therefore, the amortization time for a single speech data is 16.402 seconds, which is a significant improvement compared with the case without the SIMD technique.

Table 4: The accuracy of eight models for keyword spotting.

Method	Accuracy(%)
Proposed	**74.4**
CryptoNets	35.9
CryptoNets-1D	27.3
DPN-DNN	4.3
DPN-CNN	4.2
DPN-CNN-1D	62.9
CryptoDL	17.2
CryptoDL-1D	18.9

6.4 Comparison with the state-of-the-art Solutions

In this section, we compare the performance of our scheme with the state-of-the-art solutions on the Speech Commands dataset. We compare our CryptoℂNet with the four recent deep networks in the HE domain, i.e., DPN-CNN [48], DPN-DNN [48], CryptoNets [23], and CryptoDL [24]. The inputs of CryptoNets, CryptoDL, and DPN-CNN are two-dimensional signals. Following [48], we reshape the speech data into a two-dimensional 98×200 matrix. We choose the frame length as 25ms and the frame step as 10ms to split the speech into short-time frames. Each short-time frame constitutes each row of the matrix, and the matrix is used as the input of the three two-dimensional networks. We also converted these two-dimensional networks into a one-dimensional neural network to be fairer. Thus, we get the CryptoNets-1D, CryptoDL-1D, and DPN-CNN-1D by replacing the 2D functions of each layer in CryptoNets, CryptoDL, and DPN-CNN, respectively.

6.4.1 Accuracy. Table 4 shows the accuracy of the eight models for keyword spotting on the Speech Commands dataset. Our model has the highest accuracy of 74.4%, which is a significant improvement better than the state-of-the-art solutions. DPN-CNN and DPN-DNN did not converge during the training, thus having poor prediction accuracy. CryptoNet has an accuracy of 35.9%, which is still only half of our scheme's accuracy. CryptoDL performed worse than CryptoNets, with an accuracy of only 17.2%. It is worth noting that our proposed one-dimensional versions of DNNs and CNNs have higher accuracy than their corresponding original two-dimensional versions. For example, DPN-CNN-1D is improved by 58.7% than DPN-CNN. Thus, our experimental results show that one-dimensional neural networks are more suitable for keyword spotting than two-dimensional neural networks.

6.4.2 Robustness. We evaluate the robustness of our scheme by adding Gaussian white noises with different signal noise ratios (SNR) to the speech data. Table 5 shows the accuracies of the eight models with SNRs of 30 dB, 35 dB, and 40 dB. We can see that the accuracy of our scheme nearly remains unchanged when SNR

Table 5: Accuracy(%) of eight models under different SNR noise.

Condition	SNR(dB)	30	35	40
Accuracy(%)	Proposed	**67.3**	**73.1**	**74.3**
	CryptoNets	27.8	35.2	35.9
	CryptoNets-1D	24.5	27.6	28.7
	DPN-DNN	4.0	4.1	4.1
	DPN-CNN	3.9	3.4	3.8
	DPN-CNN-1D	54.7	62.1	62.7
	CryptoDL	11.1	14.6	16.3
	CryptoDL-1D	13.9	17.5	18.8

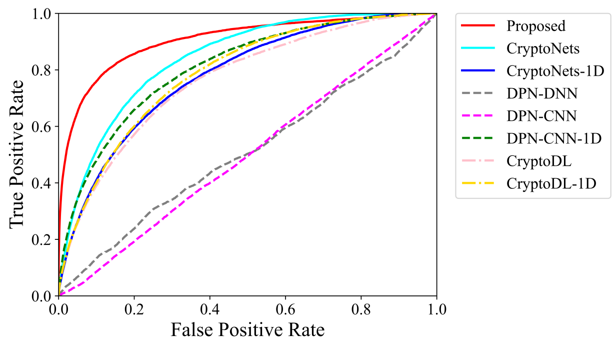

Figure 1: ROC curves of the eight models.

= 40dB. As SNR decreases, the accuracy of our scheme tends to decrease. However, even when SNR = 30dB, the accuracy of our scheme is still 67.3%, which is significantly larger than all the other compared schemes.

6.4.3 ROC Curve. Figure 1 shows the receiver operating characteristic (ROC) curves of the eight models. Both DPN-CNN and DPN-DNN achieve the worst performance, whose curves almost overlap with the diagonal. The other compared schemes perform better than DPN-CNN and DPN-DNN. However, their performances are still not satisfactory. Our scheme achieves the best performance among all the tested schemes.

6.4.4 PR Curve. Figure 2 shows the Precision-Recall (PR) curves of the eight models. As Recall increases, the precisions of the compared schemes decline rapidly with approaching the 'L' letter, indicating that the performances of these schemes are very vulnerable. In contrast, the curve of our scheme is close to a convex curve. Therefore, our scheme outperforms the compared state-of-the-art solutions.

6.4.5 Other Metrics. In Table 6, we measure the performances of our scheme and the state-of-the-art works in terms of F1 score, Recall, and Precision. Our scheme achieves the best performance among all the schemes in every metric. Our modified DPN-CNN-1D and CryptoNets are the second and third best, respectively. The experimental results show that our scheme significantly outperforms the state-of-the-art works.

Our scheme achieves an accuracy rate of over 84% in recognizing "Backward", "Sheila", "Happy", and "Six", which is much higher than the average of 74.4%. On the other hand, "Follow" and "Forward" are easily mistaken for "Four". This is because the pronunciation of these two words is relatively similar to "Four", which leads to a lower accuracy rate than the average recognition rate.

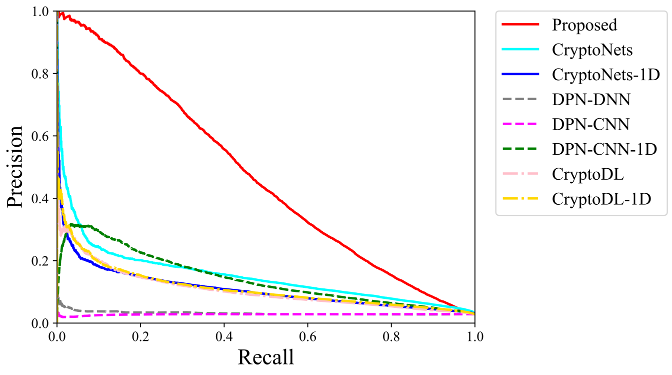

Figure 2: PR curves of the eight models.

Table 6: F1-score, Recall, and Precision of eight models.

Method	F1-score	Recall	Precision
Proposed	**73.2**	**73.1**	**73.9**
CryptoNets	33.9	34.6	36.6
CryptoNets-1D	26.6	26.7	28.8
DPN-DNN	0.4	3.1	0.2
DPN-CNN	0.3	2.8	0.3
DPN-CNN-1D	62.4	61.2	65.7
CryptoDL	11.6	14.9	16.9
CryptoDL-1D	15.5	16.5	22.2

7 CONCLUSION

In this paper, we proposed a novel scheme that enables efficient and accurate keyword spotting in the encryption domain of LHE or FHE. To fully use the multiplicative depth of the current HE scheme, our scheme uses a complex-valued CNN structure designed for keyword spotting in the HE domain. To our best knowledge, there has been no report on the complex-valued neural network in the HE domain before. We present secure implementation methods of building blocks for complex-valued CNN, including complex-valued convolution, pooling, and fully connected layers. To realize the non-linear activation function without interaction, we propose some approximation methods for complex activation functions. We propose our complex-valued convolutional network architecture for keyword spotting in the HE domain with our building blocks and approximation methods. Our scheme supports the implementation of complex-valued CNN in a batched manner, which can process multiple plaintexts simultaneously. According to our experimental results, our scheme outperforms the state-of-the-art works in terms of various metrics, including accuracy, ROC curve, and F1-score. The proposed Crypto$\mathbb{C}$Net can benefit many end-to-end audio processing applications with privacy protection requirements, such as cloud-based speech recognition and music retrieval solutions.

REFERENCES

[1] Senthildevi K. A and Chandra E. 2015. Keyword spotting system for Tamil isolated words using Multidimensional MFCC and DTW algorithm. In *2015 International Conference on Communications and Signal Processing (ICCSP)*. 0550–0554. https://doi.org/10.1109/ICCSP.2015.7322545

[2] Andreea B Alexandru, Manfred Morari, and George J Pappas. 2018. Cloud-based MPC with encrypted data. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 5014–5019.

[3] Ahmad Al Badawi, Jin Chao, Jie Lin, Chan Fook Mun, Sim Jun Jie, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, and Vijay Ramaseshan Chandrasekhar. 2018. The AlexNet Moment for Homomorphic Encryption: HCNN, the First Homomorphic CNN on Encrypted Data with GPUs. *IACR Cryptol. ePrint Arch.* 2018 (2018), 1056.

[4] Tiziano Bianchi, Alessandro Piva, and Mauro Barni. 2009. On the implementation of the discrete Fourier transform in the encrypted domain. *IEEE Transactions on Information Forensics and Security* (2009).

[5] Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. 2019. NGraph-HE2: A High-Throughput Framework for Neural Network Inference on Encrypted Data. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography* (London, United Kingdom) *(WAHC'19)*. Association for Computing Machinery, New York, NY, USA, 45–56. https://doi.org/10.1145/3338469.3358944

[6] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. 2013. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In *Cryptography and Coding*, Martijn Stam (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 45–64.

[7] Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. 2018. Fast homomorphic evaluation of deep discretized neural networks. In *Annual International Cryptology Conference*. Springer, 483–512.

[8] Zvika Brakerski. 2012. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference (Lecture Notes in Computer Science, Vol. 7417)*. Springer, 868–886. https://doi.org/10.1007/978-3-642-32009-5_50

[9] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory* 6, 3 (2014), 13:1–13:36. https://doi.org/10.1145/2633600

[10] Alon Brutzkus, Ran Gilad-Bachrach, and Oren Elisha. 2019. Low Latency Privacy Preserving Inference. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019 (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 812–821.

[11] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. 2017. Privacy-Preserving Classification on Deep Neural Network. *IACR Cryptol. ePrint Arch.* 2017 (2017), 35.

[12] Jin Chao, Ahmad Al Badawi, Balagopal Unnikrishnan, Jie Lin, Chan Fook Mun, James M. Brown, J. Peter Campbell, Michael F. Chiang, Jayashree Kalpathy-Cramer, Vijay Ramaseshan Chandrasekhar, Pavitra Krishnaswamy, and Khin Mi Mi Aung. 2019. CaRENets: Compact and Resource-Efficient CNN for Homomorphic Inference on Encrypted Medical Images. *CoRR* abs/1901.10074 (2019). arXiv:1901.10074

[13] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. Bootstrapping for Approximate Homomorphic Encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*.

[14] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. 2018. A Full RNS Variant of Approximate Homomorphic Encryption. *Springer, Cham* (2018).

[15] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*.

[16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2019. TFHE: Fast Fully Homomorphic Encryption Over the Torus. *Journal of Cryptology* 33 (04 2019). https://doi.org/10.1007/s00145-019-09319-x

[17] Hyeong-Seok Choi, Jang-Hyun Kim, Jaesung Huh, Adrian Kim, Jung-Woo Ha, and Kyogu Lee. 2018. Phase-aware speech enhancement with deep complex u-net. In *International Conference on Learning Representations*.

[18] Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, and Sungjoo Ha. 2019. Temporal convolution for real-time keyword spotting on mobile devices. *arXiv preprint arXiv:1904.03814* (2019).

[19] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril. 2019. Efficient Keyword Spotting Using Dilated Convolutions and Gating. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 6351–6355. https://doi.org/10.1109/ICASSP.2019.8683474

[20] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptol. ePrint Arch.* 2012 (2012), 144.

[21] Aravind Ganapathiraju, Jonathan Hamaker, and Joseph Picone. 2000. Hybrid SVM/HMM architectures for speech recognition. In *Sixth international conference on spoken language processing*.

[22] Craig Gentry. 2009. Fully Homomorphic Encryption Using Ideal Lattices. *Proceedings of the Annual ACM Symposium on Theory of Computing* 9, 169–178. https://doi.org/10.1145/1536414.1536440

[23] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. 2016. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016 (JMLR Workshop and Conference Proceedings, Vol. 48)*. JMLR.org, 201–210.

[24] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2017. CryptoDL: Deep Neural Networks over Encrypted Data. (2017).

[25] Yanxin Hu, Yun Liu, Shubo Lv, Mengtao Xing, Shimin Zhang, Yihui Fu, Jian Wu, Bihong Zhang, and Lei Xie. 2020. DCCRN: Deep complex convolution recurrent network for phase-aware speech enhancement. *arXiv preprint arXiv:2008.00264* (2020).

[26] Takumi Ishiyama, Takuya Suzuki, and Hayato Yamana. 2020. Highly Accurate CNN Inference Using Approximate Activation Functions over Homomorphic Encryption. In *2020 IEEE International Conference on Big Data (Big Data)*. 3989–3995. https://doi.org/10.1109/BigData50022.2020.9378372

[27] Xiaoqian Jiang, Miran Kim, Kristin E. Lauter, and Yongsoo Song. 2018. Secure Outsourced Matrix Computation and Application to Neural Networks. *IACR Cryptol. ePrint Arch.* 2018 (2018), 1041.

[28] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. Gazelle: A Low Latency Framework for Secure Neural Network Inference. (01 2018).

[29] Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. 2017. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059* (2017).

[30] Mohamed O. M. Khelifa, Yahya O. M. ElHadj, Abdellah Yousfi, and Mostafa Belkasmi. 2017. Constructing accurate and robust HMM/GMM models for an Arabic speech recognition system. *Int. J. Speech Technol.* 20, 4 (2017), 937–949. https://doi.org/10.1007/s10772-017-9456-7

[31] A. Kim. 2018. HEAAN. https://github.com/kimandrik/HEAAN

[32] M. Kim, X. Jiang, K. Lauter, E. Ismayilzada, and S. Shams. 2021. HEAR: Human Action Recognition via Neural Networks on Homomorphically Encrypted Data. (2021).

[33] James Lin, Kevin Kilgour, Dominik Roblek, and Matthew Sharifi. 2020. Training Keyword Spotters with Limited and Synthesized Speech Data. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 7474–7478. https://doi.org/10.1109/ICASSP40776.2020.9053193

[34] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On Ideal Lattices and Learning with Errors over Rings. In *Advances in Cryptology – EUROCRYPT 2010*, Henri Gilbert (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–23.

[35] Masato Mimura, Sei Ueno, Hirofumi Inaguma, Shinsuke Sakai, and Tatsuya Kawahara. 2018. Leveraging Sequence-to-Sequence Speech Synthesis for Enhancing Acoustic-to-Word Speech Recognition. In *2018 IEEE Spoken Language Technology Workshop (SLT)*. 477–484. https://doi.org/10.1109/SLT.2018.8639589

[36] Simon Mittermaier, Ludwig Kürzinger, Bernd Waschneck, and Gerhard Rigoll. 2020. Small-Footprint Keyword Spotting on Raw Audio Data with Sinc-Convolutions. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 7454–7458. https://doi.org/10.1109/ICASSP40776.2020.9053395

[37] P. Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques*.

[38] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. 2019. XONN: XNOR-based Oblivious Deep Neural Network Inference. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1501–1518. https://www.usenix.org/conference/usenixsecurity19/presentation/riazi

[39] R. L. Rivest, A. Shamir, and L. Adleman. 1977. On Digital Signatures and Public-Key Cryptosystems. (1977).

[40] Jan Stadermann and Gerhard Rigoll. 2004. A hybrid SVM/HMM acoustic modeling approach to automatic speech recognition. In *Proc. Int. Conf. on Spoken Language Processing ICSLP# 2004, Jeju Island, South Korea*.

[41] Raphael Tang and Jimmy Lin. 2018. Deep Residual Learning for Small-Footprint Keyword Spotting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5484–5488. https://doi.org/10.1109/ICASSP.2018.8462688

[42] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joo Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. 2017. Deep Complex Networks. (2017).

[43] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J. Pal. 2018. Deep Complex Networks. In *6th International Conference on Learning Representations, ICLR 2018,*.

[44] Pete Warden. 2017. Speech Commands: A public dataset for single-word speech recognition. *Dataset available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz* (2017).

[45] Minz Won, Sanghyuk Chun, Oriol Nieto, and Xavier Serrc. 2020. Data-Driven Harmonic Filters for Audio Representation Learning. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 536–540. https://doi.org/10.1109/ICASSP40776.2020.9053669

[46] Pengtao Xie, Misha Bilenko, Tom Finley, Ran Gilad-Bachrach, Kristin E. Lauter, and Michael Naehrig. 2014. Crypto-Nets: Neural Networks over Encrypted Data. *CoRR* abs/1412.6181 (2014). arXiv:1412.6181

[47] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 162–167.

[48] Shi-Xiong Zhang, Yifan Gong, and Dong Yu. 2019. Encrypted Speech Recognition Using Deep Polynomial Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019*. IEEE, 5691–5695. https://doi.org/10.1109/ICASSP.2019.8683721

[49] Peijia Zheng and Jiwu Huang. 2013. Discrete wavelet transform and data expansion reduction in homomorphic encrypted domain. *IEEE Transactions on Image Processing* 22, 6 (2013), 2455–2468.

[50] Peijia Zheng and Jiwu Huang. 2018. Efficient encrypted images filtering and transform coding with walsh-hadamard transform and parallelization. *IEEE Transactions on Image Processing* 27, 5 (2018), 2541–2556.

[51] Yimeng Zhuang, Xuankai Chang, Yanmin Qian, and Kai Yu. 2016. Unrestricted Vocabulary Keyword Spotting Using LSTM-CTC. 938–942. https://doi.org/10.21437/Interspeech.2016-753