# Efficient Privacy-Preserving Convolutional Neural Networks with CKKS-RNS for Encrypted Image Classification

Andrei Tchernykh[1,2,*], Marianne Salgado-Ramos[1]
[1]*CICESE Research Center*
Ensenada, Baja California, Mexico
[2]*Institute for System Programming*, RAS, Moscow, Russia
chernykh@cicese.mx, marianne@cicese.edu.mx

Bernardo Pulido-Gaytan[3], Horacio González-Vélez[3]
[3]*National College of Ireland*
Dublin, Ireland
{luisbernardo.pulidogaytan, horacio}@ncirl.ie

Esteban Mosckos[4]
[4]*Universidad de Buenos Aires*
Buenos Aires, Argentina
emocskos@dc.uba.ar

Mikhail Babenko[5]
[5]*North-Caucasus Federal University*
Stavropol, Russia
mgbabenko@ncfu.ru

***Abstract*—The rise of security concerns in cloud-shared infrastructures has introduced significant challenges for maintaining privacy in data processing. Although standard encryption methods provide robust protection for data at rest and during transmission, vulnerabilities arise when data must be decrypted for processing, exposing sensitive raw information to potential privacy risks. This issue is particularly pronounced in sectors governed by stringent regulatory requirements, such as healthcare, genomics, smart government, and finance, among many others, where protecting confidential data is critical. Homomorphic Encryption (HE) cryptosystems are solutions to address privacy concerns by providing encrypted data computations. HE allows a non-trustworthy third-party resource to process encrypted information without disclosure. However, the main challenge toward deploying lattice-based HE schemes in Convolutional Neural Network (CNN) models lies in overcoming the high computational costs associated with these cryptosystems. Efficient cryptographically compatible methods become imperative for designing a privacy-preserving CNN with HE (CNN-HE). This paper proposes a method to improve the performance of CNN-HE using the Residual Number System (RNS)-based Cheon-Kim-Kim Song (CKKS) HE scheme, which enables approximate arithmetic over encrypted real numbers. The CNN-HE with CKKS-RNS enables encrypted inputs to be decomposed into several parts and propagated homomorphically and independently in parallel across the model. The RNS representation enables parallel processing in our models, significantly reducing processing time. Experimental analysis on the MNIST optical character recognition benchmark dataset demonstrates that the proposed CNN-HE-RNS models reduce classification latency concerning state-of-the-art CNN-HE solutions without compromising security and accuracy.**

***Keywords—cloud security, homomorphic encryption, neural networks, polynomial approximation, privacy-preserving, residual number system decomposition***

## I. Introduction

A significant limitation to the widespread adoption of cloud computing for Machine Learning (ML) tasks involving sensitive information is the inadequate safeguarding of data privacy [1]. The access of the ML model to the raw data can create potential privacy risks because data are in the shared infrastructure, particularly in sectors where confidentiality is paramount, such as healthcare, genomics, smart government, and finance, among many others.

Conventional encryption algorithms, such as the Advanced Encryption Standard (AES), successfully protect data at rest and during transmission, preventing third parties from accessing them. However, data processing implies a decryption process for data value extraction, falling into the problem of data vulnerability. Data must be computed securely [2].

Homomorphic Encryption (HE) can solve these problems by allowing computations on encrypted data [3], [4]. HE is a form of encryption that allows algebraic operations to be performed on ciphertexts without requiring access to the secret key. Its correctness relies on the homomorphism concept, a structure-preserving transformation where two groups in different spaces can be mapped, e.g., polynomial rings, Galois fields, etc. In this case, a homomorphic function applied to ciphertexts provides the same result (after decryption) as applying the function to the original unencrypted data. In a nutshell, HE enables blind two-party non-interactive processing of sensitive data [5].

An open problem in this field is the design of privacy-preserving Convolutional Neural Network (CNN) models for classifying encrypted information using these lattice-based HE cryptosystems [6]. However, the main challenge toward deploying a privacy-preserving CNN with HE (CNN-HE) is overcoming the high computational costs associated with these HE cryptosystems. Computing ciphertexts with state-of-the-art schemes represents a slowdown of 4-6 orders of magnitude compared to performing on unencrypted data [7]. Efficient encrypted data processing is an important research area in the CNN-HE domain.

In this paper, we present a method to improve the performance of CNN-HE using the Residual Number System

(RNS)-based Cheon-Kim-Kim Song (CKKS) HE scheme, which enables approximate arithmetic over encrypted real (complex) numbers, thereby allowing for privacy-preserving computations on sensitive data in the domain of continuous values. Our CNN-HE model with CKKS-RNS enables encrypted inputs to be decomposed into several parts and then propagated homomorphically and independently across the model.

Experimental analysis on the MNIST optical character recognition benchmark dataset demonstrates that the proposed CNN-HE-RNS models reduce classification latency concerning state-of-the-art CNN-HE solutions without compromising security and accuracy. CNN-HE-RNS safety is guaranteed by using CKKS-RNS, whose security is based on the hardness of the Ring Learning with Errors (RLWE) problem. RLWE is known to be as hard as worst-case lattice problems, which are currently considered secure against quantum computer attacks. Additionally, RNS representation, a widely known variation of finite-ring isomorphism, enables parallel processing, resulting in a significant improvement in processing time.

The paper is structured as follows. Section II introduces homomorphic encryption schemes and presents the primitives of the CKKS-RNS scheme. Section III presents privacy-preserving convolutional neural networks with homomorphic encryption. Section IV discusses related work. Section V defines the experimental setup. The experimental results are presented in Section VI. Finally, we conclude in Section VII.

## II. CKKS-RNS Homomorphic Encryption Scheme

Homomorphic Encryption (HE) is a form of encryption that allows algebraic operations to be performed on encrypted data without requiring access to the secret key. The information is public without representing a risk of a data breach, as the results of the calculated operations remain encrypted. Its correctness relies on the homomorphism concept, a structure-preserving transformation where two groups in different spaces can be mapped, e.g., polynomial rings, Galois fields, etc. Therefore, a homomorphic function applied to ciphertexts provides the same result (after decryption) as using the function to the original unencrypted data. HE enables blind, two-party, non-interactive processing of sensitive data (see Figure 1).

Let $m_a$ be the message $a$ in plaintext, $sk$ a secret key for decryption, and $pk$ a public key for encryption. The corresponding ciphertext $c_a$ of $m_a$ is generated by the encryption operation $c_a = \text{Encrypt}(pk, m_a)$. Recovering the information in an additively HE from a ciphertext $c_+$ is performed by the decryption operation and the secret key as $m_+ = \text{Decrypt}(sk, c_+)$, where $c_+ = \text{Add}(c_a, c_b)$ contains the result of the homomorphic addition between $c_a$ and $c_b$. Analogously, for multiplication $c_\times = \text{Mult}(c_a, c_b)$ in a multiplicative HE. The HE schemes obtain ciphertexts $c_+$ and $c_\times$, without knowing $m_a$ and $m_b$. Ciphertexts $c_+$ and $c_\times$ cannot be computed with standard encryption without the decryption of $c_a$ and $c_b$. Each HE scheme defines a conventional public-key scheme with basic operations to generate secret and public keys, encrypt and decrypt messages, and perform homomorphic addition and multiplication operations.

Cheon-Kim-Kim Song (CKKS), also known as Homomorphic Encryption for Arithmetic of Approximate Numbers (HEAAN), is a lattice-based HE scheme whose security is based on the hardness of the Ring Learning with Errors (RLWE) problem. The CKKS scheme performs approximate arithmetic on encrypted real (complex) numbers, allowing for privacy-preserving computations on sensitive data in the domain of continuous values. This capability of the CKKS scheme is crucial for complex and demanding applications, such as privacy-preserving NN models. Given plaintext messages $m_a$ and $m_b$, it allows secure computing encryptions of approximate values of $m_a + m_b$ and $m_a m_b$ with a prefixed precision. The main characteristic of CKKS is that it treats the inserted noise of the RLWE problem as part of an error occurring during approximate computation.

In CKKS, after the selection of parameters such as degree $N$ of the polynomial ring and a modulo $q$, real numbers are encoded into plaintext polynomials. The $sk$ is a randomly generated polynomial. The $pk$ is created using $sk$, chosen parameters, and some randomness. The plaintext polynomial is encrypted using the $pk$ and noise to increase security. After operations, rescaling is performed to reduce noise and preserve the ciphertext. The ciphertext is decrypted using $sk$ to obtain an approximated version of the original plaintext polynomial decoded into a real or complex resulting number.

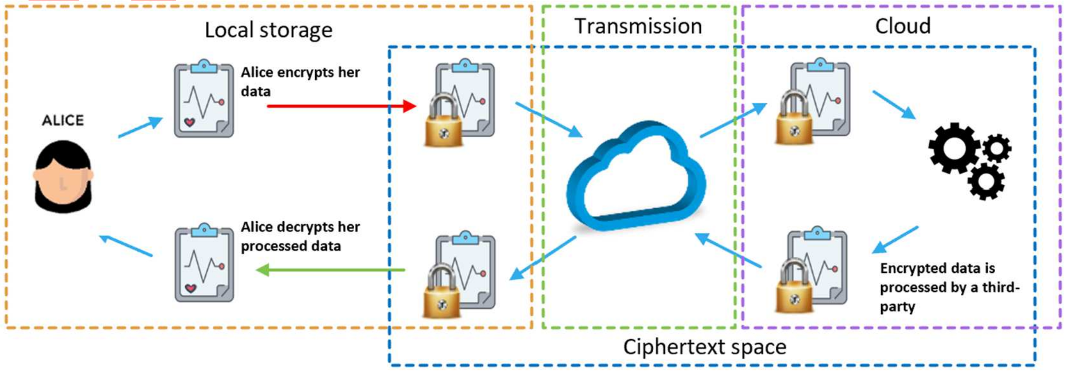

Local storage
Transmission
Cloud
ALICE
Alice encrypts her data
Alice decrypts her processed data
Encrypted data is processed by a third-party
Ciphertext space

Fig. 1. Overview of privacy-preserving processing in a cloud environment

Let the polynomial ring for a power-of-two $N$ be $R = \mathbb{Z}[X]/(X^N + 1)$ and $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ be a modulo-$q$ residue ring of $R$. Polynomial coefficients of $R_q$ are bounded by the modulo $q$ and the degree of polynomials $X^N + 1$. Let $L$ be a level parameter that indicates the maximum multiplicative depth. Modulo $q = q_0 \cdot q_1 \cdot \ldots \cdot q_L$ is defined as the product of co-prime moduli $q_0, \ldots q_L$, where $q_\ell = 2^\ell \cdot q_0$ for $1 \leq \ell \leq L$.

The distribution $\chi_{\text{key}} = \text{HW}(h)$ outputs a polynomial from $R_q$ of $\{\pm 1\}$-coefficient having $h$ non-zero coefficients, where $\text{HW}(h)$ denotes the set of signed binary vectors in $\{\pm 1\}^N$ whose Hamming weight is $h \in Z_+$. $\chi_{\text{enc}}$ and $\chi_{\text{err}}$ denote discrete Gaussian distributions with some predefined standard deviation. $U(R_q)$ refers to a uniform distribution over the ring $R_q$. Moreover, for $a = \sum_{i=0}^{N-1} a_i X^i \in \mathbb{R}[X]/(X^N + 1)$, then $\lfloor a \rceil = \sum_{i=0}^{N-1} \lfloor a_i \rceil X^i \in R$, where $\lfloor \cdot \rceil$ returns the nearest integer of a real-number input, rounding upwards in case of a tie [8].

CKKS encodes real values using a called canonical embedding $\tau$: $\mathbb{R}[X]/(X^N + 1) \to \mathbb{C}^{N/2}$, where a plaintext vector $\vec{m} = (m_0, \ldots, m_{N/2})$ is transformed into $\tau^{-1}(\vec{m}) \in \mathbb{R}[X]/(X^N + 1)$ and then rounded to an integer-coefficient polynomial using a scaling factor $\Delta$, i.e., $\lfloor \Delta \cdot \tau^{-1}(\vec{m}) \rceil$. The parameter $\Delta$ affects the accuracy of the computation in CKKS.

Let us discuss the main primitives of the CKKS scheme:

- KeyGen $(N, q, L) \to sk, pk, ek$: Sets $sk = (1, s)$, where $s \leftarrow \chi_{\text{key}}$. Sets $pk = (b, a) \in R_{q_L}^2$, where $b = -as + e \pmod{q_L}$, $a \leftarrow U(R_{q_L})$, and $e \leftarrow \chi_{\text{err}}$. The evaluation key is set as $ek = (b', a') \in R_{q_L^2}^2$, where $b' = -a's + e' + q_L s^2 \pmod{q_L^2}$, $a' \leftarrow U(R_{q_L^2})$ and $e' \leftarrow \chi_{\text{err}}$.
- Encrypt $(\vec{m}, \Delta, pk) \to c$: For a plaintext vector of real (complex) numbers $\vec{m}$, it encodes $m = \lfloor \Delta \cdot \tau^{-1}(\vec{m}) \rceil \in R$, and provides the ciphertext $c = v \cdot pk + (m + e_0, e_1) \pmod{q_L}$, where $v \leftarrow \chi_{\text{enc}}$ and $e_0, e_1 \leftarrow \chi_{\text{err}}$.
- Decrypt $(c, \Delta, sk) \to \vec{m}$: For a ciphertext $c = (c_0, c_1) \in R_{q_\ell}^2$, decodes the message as $m = c_0 + c_1 \cdot s \pmod{q_\ell}$, and outputs a plaintext vector $\vec{m} = \Delta^{-1} \cdot \tau(m)$.
- Add $(c_1, c_2) \to c_+$: Add two ciphertexts $c_1, c_2 \in R_{q_\ell}^2$. It returns ciphertext $c_+ = c_1 \oplus c_2 = c_1 + c_2 \pmod{q_\ell}$.
- Mult $(c_1, c_2, ek) \to c_\times$: For two ciphertexts $c_1 = (c_{1,0}, c_{1,1})$, $c_2 = (c_{2,0}, c_{2,1}) \in R_{q_\ell}^2$, let $(d_0, d_1, d_2) = (c_{1,0}c_{2,0},\ c_{1,0}c_{2,1} + c_{1,1}c_{2,0},\ c_{1,1}c_{2,1})$. It returns ciphertext $c_\times = c_1 \otimes c_2 = (d_0, d_1) + \lfloor q_L^{-1} \cdot d_2 \cdot ek \rceil \pmod{q_\ell}$.
- Rot $(c, r) \to c'$: Given an encryption $c$ of $\vec{m} = (m_0, \ldots, m_{N/2})$, outputs $c'$ that encrypts the left-rotated vector $\vec{m} = (m_r, \ldots, m_{N/2}, m_0, \ldots, m_{r-1})$ by $r$ positions.

Because each $\vec{m}$ is scaled, the plaintext of $c \leftarrow$ Mult $(c_1, c_2)$ is $\Delta \cdot m_1 m_2$, which results in the exponential growth of plaintexts. To deal with such a problem, CKKS introduces the so-called rescaling procedure:

- Resc $(c) \to c'$: For a ciphertext $c \in R_{q_\ell}^2$, outputs $c' = \lfloor q_{\ell'}/q_\ell \rceil \cdot c \pmod{q_{\ell'}}$.\

Since the CKKS scheme requires somewhat large integers, the original implementation relies on a multi-precision library, which leads to higher computational complexity. To mitigate this complexity, a variant known as the Residue Number System (RNS)-CKKS scheme [9] was introduced. The application of RNS allows homomorphic operations to be performed in parallel and reduces the complexity of calculations, leading to a significant improvement in processing time.

In CKKS-RNS, large integers are decomposed into several smaller integers, such that addition and multiplication operations on the original large integers are performed as component-wise operations on their smaller counterparts (see Figure 2). In this paper, RNS-CKKS was employed as the underlying HE cryptosystem.

For more detailed information and additional considerations on the CKKS and CKKS-RNS schemes, including the correctness and security analysis, refer to [8] and [9], respectively.

## III. Privacy-Preserving CNN-HE with CKKS-RNS

This section explores the design of privacy-preserving NN-HE as a natural extension of conventional non-encrypted NN models.

### *A. Homomorphic neuron*

The neuron is the fundamental unit in an artificial NN, processing inputs through weighted transformations and producing outputs via a nonlinear activation function. A NN architecture defines a set of neurons organized in layers with connections between them.

The methodology for designing a CNN-HE model involves applying HE to the inputs and homomorphically propagating the data across the network. However, its implementation faces limitations, as not all functions for processing NN have direct homomorphic counterparts [10]. To ensure secure computation, the internal structure of the model must be adapted, typically by replacing non-homomorphic functions with suitable approximations.

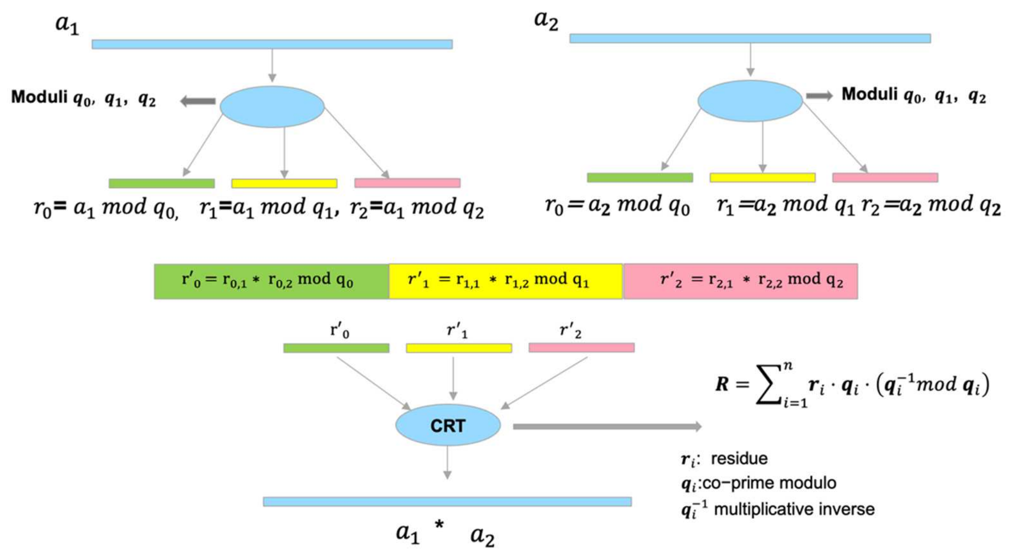
Let us define the homomorphic neuron in a CNN-HE as

$$\bar{y} \leftarrow \ddot{f}\left(\sum_{i=1}^{n} (\bar{w}_i \otimes c_i) \dotplus \bar{\beta}\right), \qquad (1)$$

where $n$ denotes the number of ciphertext inputs $c_i$ with weights $\bar{w}_i$. Bias $\bar{\beta}$ is also a ciphertext, and the activation function $\ddot{f}$ is a polynomial approximation that only consists of homomorphic operations $\oplus$ and $\otimes$ (see Section II). $\bar{y}$ is the encrypted output of the neuron.

The weighted sum in (1) consists of additions and multiplications between the encrypted inputs and their corresponding synaptic weights, enabling its homomorphic computation.

However, an open problem in the CNN-HE domain is the definition of cryptographically compatible nonlinear components. State-of-the-art activation functions are not polynomials and use operations that HE does not support. To overcome this limitation, we adopt polynomial self-learning activation functions [10], [11], [12].



$a_1$
$a_2$
Moduli $q_0, q_1, q_2$
Moduli $q_0, q_1, q_2$
$r_0 = a_1 \bmod q_0$, $r_1 = a_1 \bmod q_1$, $r_2 = a_1 \bmod q_2$
$r_0 = a_2 \bmod q_0$ $r_1 = a_2 \bmod q_1$ $r_2 = a_2 \bmod q_2$
$r'_0 = r_{0,1} * r_{0,2} \bmod q_0$
$r'_1 = r_{1,1} * r_{1,2} \bmod q_1$
$r'_2 = r_{2,1} * r_{2,2} \bmod q_2$
$r'_0$
$r'_1$
$r'_2$
CRT
$R = \sum_{i=1}^{n} r_i \cdot q_i \cdot (q_i^{-1} \bmod q_i)$
$r_i$: residue
$q_i$: co-prime modulo
$q_i^{-1}$ multiplicative inverse
$a_1 * a_2$

Fig. 2. Residue number system decomposition

## B. *Self-learning activation functions*

Based on the definition of a homomorphic neuron outlined in Section III.A, we adopt the Self-Learning Activation Functions (SLAF) approach [11], approximating the nonlinear activation function at each neuron independently using a polynomial with trainable coefficients. That is, CNN-HE activation functions are defined as

$$\ddot{f}_k = a_0^k \oplus a_1^k x \oplus a_2^k x^2 \oplus \ldots \oplus a_n^k x^n \tag{2}$$

where $a_0^k, a_1^k, \ldots, a_n^k$ denote the trainable coefficients of the polynomial $\ddot{f}_k$ at neuron $k$. Our CNN-HE considers $n+1$ trainable polynomial coefficients $a_0, a_1, \ldots, a_n$ initialized by zero.

The homomorphically computable SLAF allows us to find an adequate function shape using knowledge from the training data [13]. SLAF coefficients are learned together with the model parameters using the backpropagation learning algorithm and gradient descent optimization method. Thus, polynomial SLAFs are adapted to the specific problem, network structure, hyperparameters, and dataset.

Multiple studies indicate that NN models with polynomial activations have the same representational power as their non-polynomial analogous [14], [15]. These polynomial activations enhance latency without compromising accuracy [16]. Additionally, a CNN-HE with SLAF can approximate any continuous activation [11] [17].

## C. *Errors*

CNN-HE implementation using approximate HE schemes, such as CKKS and CKKS-RNS, can compromise the correctness of the results. This section examines the errors introduced during homomorphic processing to elucidate the accuracy loss in CNN-HE models compared to their non-homomorphic counterparts.

Encoding real (complex) numbers into a polynomial ring with integer coefficients can lead to an incorrect result. An error can occur in numbers in the zero neighborhood, even without adding noise. Converting a value vector to polynomial $m(X)$ can lead to an error in calculation. Let us consider the CKKS [8] scheme to illustrate this concern:

Let the $M$-th cyclotomic polynomial with $M = 8$ (i.e. $\Phi_8(X) = X^4 + 1$), $\Delta = 64$, and $T = \{\xi_8, \xi_8^3\}$ for the root of unity $\xi_8 = \exp(2\pi i/8)$. For vector $z = (0.1, -0.01)$, its corresponding real polynomial is $-0.039X^3 + 0.039X + 0.045$ according to the interpolation polynomial in the Lagrange form for a given set of points $(\xi, 0.1)$, $(\xi^3, -0.01)$, $(\bar{\xi}, \overline{0.1})$, $(\overline{\xi^3}, \overline{-0.01})$, where $\overline{a+bi} = a - bi$, $a, b \in \mathbb{R}$ and $i^2 = -1$. Therefore, the encoding algorithm output is $m(X) = -2X^3 + 2X + 3$. Note that $64^{-1} \cdot \left(m(\xi_8), m(\xi_8^3)\right) \approx (0.09107, 0.00268)$ is approximated to the vector $z$ with high precision. This example shows that the encoding number $-0.01$ turned into $0.00268$ when decoded. The number obtained during decoding differs in the value and sign, i.e., it does not carry any information about the initial number. Moreover, increasing $\Delta$ allows to reduce the absolute value. Consequently, an incorrect result using a CNN-HE model is highly probable when the input data are normalized, i.e., the values are compressed up to the interval [0,1]. This error leads to incorrect results when using unstable algorithms [18].

On the other hand, calculation errors can arise due to the polynomial approximation of nonlinear activation functions. For instance, let us consider $\mathrm{ReLU}(x) = \max(x, 0) = x\,(\mathrm{sign}(x) + 1)$. The function $\max(a, b)$ returns the maximum value of $a$ and $b$. When we calculate $\mathrm{ReLU}(x)$ for $x < 0$ over HE-encrypted data using $\mathrm{sign}(x)$ algorithm presented by Cheon et al. [19], the function will be greater than zero. Thus, the implementation of CNN-HE should be considered rounding errors that may occur.

## IV. Related Work

This paper aims to be a step towards bringing closer the CNN and HE cryptography fields. However, the main challenge toward deploying HE in cognitive models is overcoming the high computational costs associated with these cryptosystems. Computing over ciphertexts with state-of-the-art schemes such as CKKS and CKKS-RNS represents a slowdown of 4-6 orders of magnitude compared to performing the same computations on unencrypted data [7]. While several approaches have been proposed in the literature to accelerate the homomorphic processing of these solutions, they remain inefficient in practice. Consequently, designing efficient cryptographically compatible methods to optimize privacy-preserving CNN-HE models remains an open challenge.

CryptoNets, proposed by Dowlin et al. [20], represents the first model that addresses the challenge of achieving a blind, non-interactive classification. It uses the leveled YASHE [21] for inputs and propagates signals across the network homomorphically. However, its performance is limited due to the square function being used as an activation function, computational overhead, and the insecure YASHE scheme [22]. Several subsequent works in the literature focus on improving its constraints. Chabanne et al. [23] enhanced CryptoNets' performance by employing a polynomial approximation of the ReLU activation function and using a batch normalization layer before each activation layer. Chou et al. [24] improved encrypted inference speed through pruning and quantization methods that leverage sparse representations in the underlying Brakerski/Fan-Vercauteren (BFV) [25], achieving competitive accuracy while reducing computational time.

Bourse et al. [26] presented a Discretized Neural Network (DiNN) for inference over encrypted data, whose complexity is linear in the network size. However, it incurs high overhead and low accuracy because each neuron output is refreshed through the bootstrapping procedure featured by the Fully Homomorphic Encryption scheme over the Torus (TFHE). Sanyal et al. [27] proposed a privacy-preserving Binary Neural Network (BNN) with a TFHE scheme built upon the DiNN approach, which evaluates arithmetic operations as a composition of binary gates. The approach implements sparsification techniques and algorithmic tools to speed up and parallelize ciphertext computation but faces significant latency, requiring up to 37 hours for single predictions.

Van Elsloo et al. [28] proposed SEALion, an extensible framework built upon CryptoNets that automates encryption parameter selection for BFV, improving both latency and encrypted inference. Hesamifard et al. [29] developed CryptoDL, an NN-HE based on CryptoNets [20], [23]. CryptoDL replaces nonlinear activations with HE-friendly low-degree approximations. Liao et al. [30] extended this by implementing a CryptoDL-based NN-HE for encrypted sensor data; they approximated Tanh, ReLU, and Swish to generate cryptographically computable activations. Brutzkus et al. [31] proposed Low-Latency (Lo-La) CryptoNets to improve latency and memory usage. Lo-La encrypts entire layers as a single message with BFV and uses different matrix-vector multiplication implementations, improving latency but remaining constrained by message dimension.

Boemer et al. [32] presented nGraph-HE, an extension of the Intel nGraph compiler to deploy NN-HEs. It incorporates a privacy-preserving abstraction layer, enabling HE-aware optimizations at compile- and run-time. Jiang et al. [33] proposed the E2DM framework for arithmetic on encrypted matrices, enabling Single Instruction Multiple Data (SIMD) computations. Their CryptoNets implementation processes ten likelihoods of 64 MNIST images in 1.69 seconds. Falcetta and Roveri [34] developed a privacy-preserving LeNet-1, focusing on security parameter selection and approximation of nonlinear layers.

Badawi et al. [35] presented an efficient GPU-based BFV implementation for NN-HE, significantly accelerating the classification process while maintaining accuracy; it classifies MNIST in 2% of the time CryptoNets takes. Lee et al. [36] deployed a ResNet-20 using RNS-CKKS with bootstrapping. The authors homomorphically evaluate the NN-HE with 383 CIFAR-10 images and plaintext model parameters. However, inferring one image takes about 3 hours due to the more than a thousand bootstrapping functions and the high-degree minimax composite polynomials. Pulido-Gaytan and Tchernykh [11] proposed non-interactive privacy-preserving SLAF-based CNN-HE models to classify CKKS-encrypted data with improved accuracy and performance. The authors show that a self-learning mechanism can achieve the same accuracy of 99.38% as a non-polynomial ReLU over non-homomorphic CNNs and increase accuracy and higher performance than the state-of-the-art CNN-HE CryptoNets.

Table 1 summarizes the main features of the state-of-the-art NN-HE models, focusing on implementation details, latency (Lat), and accuracy (Acc). The GPU (Graphic Processing Unit) column denotes the use of GPU for hardware acceleration. The 2-arch (2-architecture) column indicates using a dual-architecture strategy by collapsing adjacent linear layers during the evaluation process.

## V. Experimental Setup

This section describes the evaluation method, developing tools, dataset, security parameter settings, and model architectures.

### *A. Tools*

The CKKS-RNS scheme, homomorphic operations, and privacy-preserving CNN-HE-RNS models are implemented using PyTorch [15], a widely used open-source library for deep learning and tensor processing, and the open-source SEAL v3.5.6 [16] through the Python TenSEAL library [17]. Given that the current version of TenSEAL lacks support for RNS composition and decomposition, a custom CKKS-RNS implementation has been developed. The experimental evaluation is performed on a server Express x3650 M4 with Intel(R) Xeon(R) CPU E5-2650v2 95W at 2.6GHz, 64 GB. The 64-bit server OS is Ubuntu 18.04.6.

### *B. Security settings*

The noise budget, ciphertext and plaintext size, scheme performance, multiplicative depth, and security level depend on the security parameter settings. We adopt the security settings specified in the HE standard [37]. While the multiplicative depth of the mathematical convolution is equal to one, the multiplicative depth of the polynomial evaluation is equal to the binary logarithm of the polynomial plus one. In Table II, we show such security settings for the CKKS-RNS scheme. The security level $\lambda = 128$ bits guarantees that an adversary needs to perform $2^{128}$ elementary operations to break the scheme with a probability one.

TABLE I. STATE-OF-THE-ART PRIVACY-PRESERVING NN-HE

| Year | Model | Dataset | Performance | | GPU | 2-arch | Ref |
|---|---|---|---|---|---|---|---|
| | | | Lat (s) | Acc (%) | | | |
| 2016 | CryptoNets | MNIST | 250 | 98.95 | | ● | [20] |
| 2017 | Chabanne-NN | MNIST | NR* | 97.95 | | | [23] |
| | | | NR* | 99.28 | | | |
| 2018 | F-CryptoNets | MNIST | 39.1 | 98.70 | | | [24] |
| | | CIFAR-10 | 22372 | 76.72 | | | |
| 2018 | FHE-DiNN100 | MNIST | 1.65 | 96.35 | | | [26] |
| 2018 | TAPAS | MNIST | 37 [hrs.] | 98.60 | | | [27] |
| 2019 | SEALion | MNIST | 60 | 98.91 | | | [28] |
| 2019 | CryptoDL | MNIST | 148.97 | 98.52 | | | [29] |
| | | | 320 | 99.25 | | | |
| 2019 | Lo-La | MNIST | 0.29 | 96.92 | | | [31] |
| | | | 2.20 | 98.95 | | ● | |
| | | CIFAR-10 | 730 | 74.10 | | | |
| 2019 | nGraph-HE | MNIST | 16.72 | 98.95 | | ● | [32] |
| | | CIFAR-10 | 1651 | 62.20 | | | |
| 2019 | E2DM | MNIST | 1.69 | 98.10 | | ● | [33] |
| 2021 | HCNN | MNIST | 5.16 | 99.00 | ● | | [35] |
| | | CIFAR-10 | 304.43 | 77.55 | | | |
| 2022 | LeNet-HE | MNIST | 138 | 98.18 | | | [34] |
| 2022 | RNS-CKKS-NN | CIFAR-10 | 10602 | 92.43** | ● | | [36] |
| 2024 | CNN-HE-SLAF | MNIST | 3.13 | 98.22 | | | [11] |
| | | | 39.84 | 99.21 | | ● | |

NR*: It does not provide results over encrypted data. Therefore, we cannot provide a comparison w.r.t to the performance measures.

**: Classification accuracy with 383 encrypted images.

## C. Dataset

The Modified National Institute of Standards and Technology (MNIST) database is a standard dataset widely used in the literature [38]. It consists of 60,000 grayscale images of handwritten digits. Each image is a 28x28 pixel array, where the value of each pixel is a positive integer in the range [0, 255]. The MNIST training set includes 50,000 examples. The remaining 10,000 images represent the testing set. While MNIST is arguably a simple dataset, it has remained the standard benchmark for homomorphic inference tasks [29], [39].

TABLE II. CKKS-RNS SECURITY SETTINGS

| *Parameter* | *Value* |
|---|---|
| $\lambda$ | 128 |
| $N$ | $2^{14}$ |
| $\Delta$ | $2^{26}$ |
| $\log q$ | 366 |
| $L$ | 13 |
| $q$ | $[40, 26, \ldots, 26, 40]$ |

## D. Architectures

We study two CNN architectures: 1-convolutional (CNN1) and 2-convolutional (CNN2). To provide a direct comparison concerning state-of-the-art solutions, we adopt the CNN1-HE-SLAF and CNN2-HE-SLAF [11] architectures for CNN1 and CNN2, which in turn are variants of CryptoNets [20], Faster-CryptoNets [24], CryptoDL [29], Lo-La [31], HCNN [35], SEALion [28], and others. In the CNN-HE-SLAF framework, the model is trained with the original ReLU activation function, weights are fixed, SLAFs substitute activations, and CNN is shortly re-trained to learn customized polynomial approximation coefficients.

CNN1 and CNN2 architectures are depicted in Figures 3 and 4, respectively. Additionally, Figure 5 illustrates their adaptation to RNS.

CNN1, with a single convolutional layer and two dense layers, is a variant of Lo-La [31]. In contrast to the Lo-La approach, the CNN1 model incorporates approximated activation functions after the convolutional and the first dense layers.

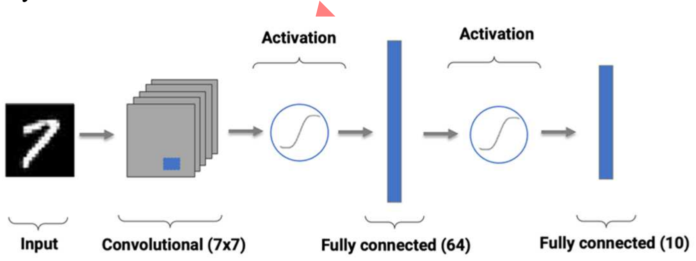


Fig. 3. CNN1 with a single convolutional layer. A gray element denotes a convolutional layer. Circular elements after the convolutional and first fully connected layers denote activation functions. Blue elements represent dense layers.

CNN2 is a CryptoNets-based model architecture with two convolutional layers. It incorporates a batch normalization layer before each activation to encourage the activation inputs to fit in the approximated interval. The normalization layer transforms them into normal distribution inputs with zero mean and unit variance, which reduces the overall approximation error, prevents the generalization of higher feature values, and indirectly provides smaller weights for HE processing.

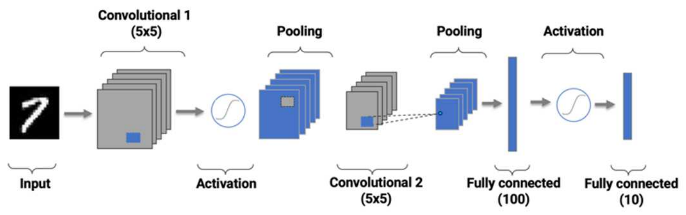
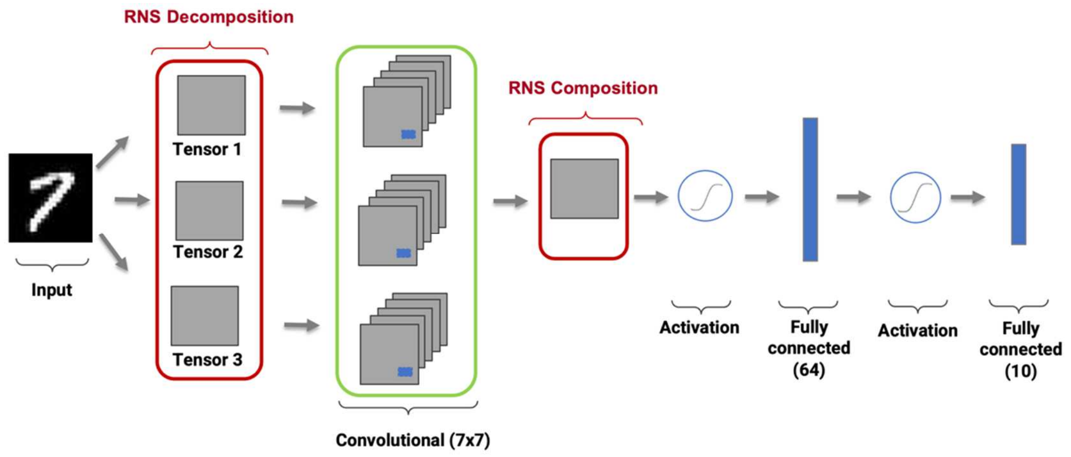
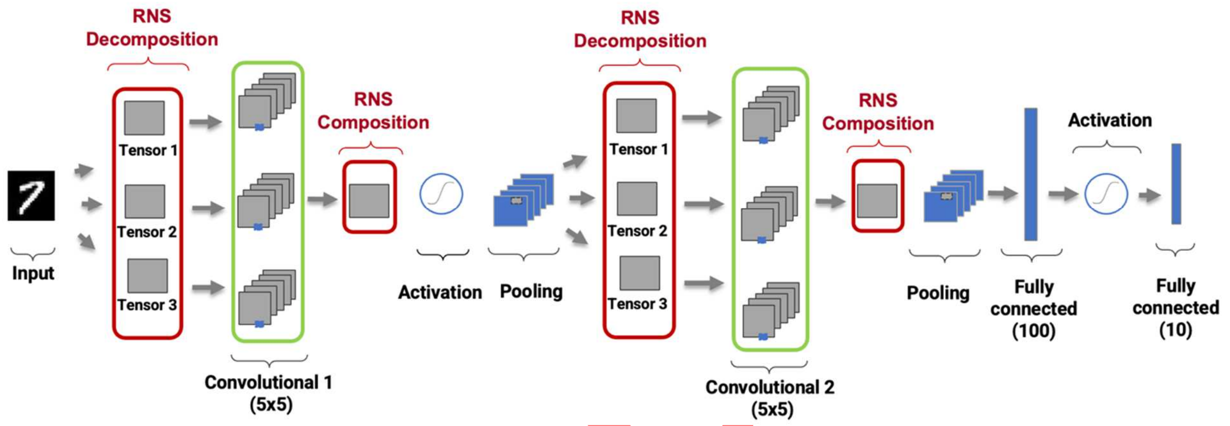


Fig. 4. CNN2: a CryptoNets-based network with two convolutional layers. Circular elements denote activation functions.

CNN1-RNS and CNN2-RNS architectures denote the adaptation of CNN1 and CNN2 models to RNS processing. As shown in Figure 5a, CNN1-RNS decomposes the input image into multiple tensors using RNS representation. The convolutional layer then processes these tensors in parallel, optimizing the handling of ciphertexts. Subsequently, the tensors are reverted to their original representation. The activation functions and dense layers are processed similarly to standard CNN1. Similarly, the CNN2-RNS architecture employs RNS to decompose the signal to process convolutional layers, with the tensors being reassembled following the convolution (see Figure 5b).



a) CNN1-RNS



b) CNN2-RNS

Fig. 5. CNN-RNS architectures. a) CNN1-RNS with a single convolutional layer. b) CNN2-RNS architecture with two convolutional layers. They employ RNS to decompose the signal for the processing of convolutional layers, with the tensors being reassembled following the convolution.

The prevailing approaches use pre-trained models to mitigate the computational overhead associated with the training phase in CNN-HE modeling, allowing a trade-off between complexity and accuracy. That is, the training of our models is performed over unencrypted data, and the evaluation is kept on the encrypted data. State-of-the-art solutions such as CryptoNets, Chabanne-NN, F-CryptoNets, FHE-DiNN100, TAPAS, SEALion, CryptoDL, Lo-La, nGraph-HE, E2DM, HCNN, LeNet-HE, RNS-CKKS-NN, CNN-HE-SLAF (see Table I), also adopt this practice [6].

The networks are trained with Stochastic Gradient Descent (SGD) with a momentum of 0.9. We utilize a batch size of 64 and a cross-entropy loss function. Thirty epochs better train models for testing accuracy. For the learning rate, we apply the 1-cycle policy [40], also called the super-convergence phenomenon. It uses one round of an increasing and decreasing learning rate, in which the maximum learning rate serves as a regularizer. We adopt the Kaiming weight initialization for convolutional layers without dropout to obtain the initial synaptic weights.

## VI. Experimental Analysis

This section presents the experimental results of the proposed privacy-preserving CNN-HE-RNS model evaluation and provides a comprehensive comparison concerning state-of-the-art solutions. The assessment of the privacy-preserving solutions is centered around their accuracy and latency.

We adapt the CNN-HE-SLAF approach for RNS to reduce processing time without compromising inference accuracy. Our models take advantage of CKKS-RNS to optimize encrypted operations in terms of latency. To compare CNN-HE-SLAF models and solutions with RNS, we analyze their differences and similarities in the training and testing phases. In all models, both inputs and weights are encrypted before testing.

### A. *CNN1-HE-RNS*

We evaluated the performance of privacy-preserving CNN1-HE and CNN1-HE-RNS models. Table III presents the models' latency (Lat) and accuracy (Acc). Results for ciphertext inputs are reported in the CNN1-HE and CNN1-HE-RNS columns. Latency refers to the computational time required to process a single encrypted classification request.

TABLE III. PERFORMANCE OF CNN1-HE AND CNN1-HE-RNS

| Model | Training Acc (%) | Lat (s) | | | Acc (%) |
|---|---|---|---|---|---|
| | | Min | Max | Avg | |
| CNN1-HE | 99.442 | 3.12 | 4.02 | 3.56 | 98.22 |
| CNN1-HE-RNS | | 1.73 | 2.89 | 2.27 | 98.22 |

We can see that the CNN1-HE-RNS implementation with a three-degree polynomial SLAF and three co-prime moduli achieves the same testing accuracy as CNN1-HE, indicating that RNS representation does not compromise the model accuracy. Additionally, experimental results show that the CNN1-HE-RNS model reduces classification latency concerning CNN1-HE. These results highlight the efficiency of CNN1-HE-RNS in terms of processing time without compromising accuracy, suggesting promising potential for practical applications.

CNN1-HE model has an average processing time of 3.56 seconds, ranging from 3.12 to 4.02. In contrast, the CNN1-HE-RNS model shows a significant improvement, with an average processing time of 2.27 seconds, ranging from 1.73 to 2.89, representing a speed-up of 36.24% on average time.

Non-positional RNS representation allows a large number to be decomposed into smaller residues that can be processed independently, accelerating convolution layers. Moreover, using smaller residuals minimizes the accumulation of numerical errors, preserving the accuracy without the need for additional calculations for error correction. These optimizations contribute to a significant decrease in processing time.

We analyze the CNN1-HE-RNS performance across various modulo configurations. This analysis is critical for understanding how model latency is affected by an increase in the number of moduli. We use the co-prime generation tool provided by SEAL, where given a list of lengths of at most 60 bits, also known as a moduli chain, a set of co-primes of those lengths is generated (see Table II). Table IV provides the average processing times obtained for each configuration.

TABLE IV. PERFORMANCE OF CNN1-HE-RNS WITH MODULO CONFIGURATIONS

| Moduli chain length | Lat (s) |
|---|---|
| 3 | 2.27 |
| 4 | 2.02 |
| 5 | 1.98 |
| 6 | 1.89 |
| 7 | 1.85 |
| 8 | 1.74 |
| 9 | 1.67 |
| 10 | 1.74 |

We can see that average model latency tends to reduce as the number of moduli increases from three to nine, with the minimum latency observed at nine moduli. However, when the number of modules is increased to ten, the average processing time rises to 1.74 seconds, indicating the presence of an optimal value for the number of modules; beyond this threshold, additional modules do not necessarily enhance latency.

## B. *CNN2-HE-RNS*

Table V presents the performance of the CNN2-HE and CNN2-HE-RNS models. Our CNN2-HE-RNS solution using three-degree polynomial activations and three co-prime moduli achieves the same testing accuracy as a non-RNS homomorphic model in the ciphertext space. These results show the remarkable capability of RNS variants to yield comparable accuracy to CNN-HE solutions. It confirms that CNN-HE-RNS models have the same representative ability as their non-RNS counterparts.

TABLE V. PERFORMANCE OF CNN2-HE AND CNN2-HE-RNS

| Model | Training Acc (%) | Lat (s) | | | Acc (%) |
|---|---|---|---|---|---|
| | | Min | Max | Avg | |
| CNN2-HE | 99.338 | 25.62 | 40.21 | 39.91 | 99.21 |
| CNN2-HE-RNS | | 21.91 | 28.35 | 23.67 | 99.21 |

Comparing performance, the CNN2-HE model presents an average processing time of 39.91 seconds, ranging from 25.62 to 40.21 seconds. On the other hand, our CNN2-HE-RNS model shows a significant improvement, with an average processing time of 23.67 seconds, ranging from 21.91 to 28.35 seconds, representing a speed-up of 40.69% on average time. The results demonstrate that CNN2-HE-RNS classifies the MNIST optical character recognition benchmark dataset 10.57 times faster than the state-of-the-art CNN-HE CryptoNets with better accuracy (see Table I). Therefore, the application of RNS to privacy-preserving CNN-HE models not only preserves the accuracy but also enhances processing latency, suggesting promising potential for applications requiring high computational efficiency.

We evaluate the performance of the CNN2-HE-RNS model through different modulo configurations, i.e., we study the impact of increasing the number of moduli on model latency. Table VI shows the average processing times obtained for each configuration.

The latency evaluation of CNN2-HE-RNS under different moduli configurations shows a reduction in the average processing time with an increasing number of moduli, reaching a minimum of 22.46 seconds with nine modules. However, adding the tenth moduli resulted in a slight increase in the average processing time to 22.51 seconds, indicating that nine moduli represent the optimal setting for minimizing latency.

The experimental results demonstrate that the integration of RNS representation significantly enhances the efficiency of CNN-HE models without sacrificing encrypted classification accuracy. Furthermore, parallelization is identified as a critical strategy for optimizing overall model efficiency. These findings offer valuable insights for the practical implementation of CNN-

HE models, ensuring an adequate trade-off between security, accuracy, and computational complexity.

TABLE VI. PERFORMANCE OF CNN2-HE-RNS WITH MODULO CONFIGURATIONS

| Moduli chain length | Lat (s) |
|---|---|
| 1 | 39.91 |
| 3 | 23.67 |
| 4 | 23.39 |
| 5 | 23.12 |
| 6 | 22.76 |
| 7 | 22.54 |
| 8 | 22.49 |
| 9 | 22.46 |
| 10 | 22.51 |

## VII. CONCLUSION

Privacy-preserving techniques, such as HE, offer a solution for addressing the processing of confidential data while it remains encrypted. Nonetheless, the computational overhead is the biggest challenge to the widespread adoption of HE. Optimization techniques are required to improve performance in various domains, such as cloud CNN-HE modeling. Although protecting the privacy of sensitive data is essential, CNN-HE models must also offer acceptable time complexity.

In this paper, we propose a method to enhance the performance of CNN-HE models by utilizing the CKKS-RNS HE scheme, enabling efficient encrypted image classification. Our CNN-HE-RNS enables encrypted inputs to be decomposed into several parts and then propagated homomorphically and independently in parallel across the model. The RNS representation enables parallel processing in our models, significantly reducing processing time. Experimental analysis on the MNIST optical character recognition benchmark dataset demonstrates that the proposed CNN-HE-RNS models reduce classification latency concerning state-of-the-art CNN-HE solutions without compromising security and accuracy.

The proposed CNN-HE-RNS models yield better performance than state-of-the-art solutions. The CKKS-RNS scheme supports optimization. Nonetheless, several lines of work in the encrypted classification domain remain to be addressed in future work. It is essential to apply the state-of-the-art practical methods of NN hardware acceleration to proposed models: GPU, Tensor Processing Unit (TPU), Field Programmable Gate Array (FPGA), Application-Specific Integrated Circuit (ASIC), etc.

Moreover, the list of potential applications is broad due to the applied nature of discussed real-world problems and the privacy benefits of CNN-HE-RNS solutions. In future work, it is essential to explore the applicability of proposed models for sensitive domains such as medical image classification.

## ACKNOWLEDGMENT

This research was funded by the European Commission under grant 101084013 (DIGITAL4Business) and the Ministry of Science and Higher Education of the Russian Federation under grant number 075-15-2022-294.

## REFERENCES

[1] M. Babenko, A. Tchernykh, B. Pulido-Gaytan, J. M. Cortés-Mendoza, E. Shiryaev, and E. Golimblevskaia, "RRNS base extension error-correcting code for performance optimization of scalable reliable distributed cloud data storage," in 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2021, pp. 548–553. doi: 10.1109/IPDPSW52791.2021.00087.

[2] A. Tchernykh, M. Babenko, A. Avetisyan, and A. Yu. Drozdov, "En-AR-PRNS: Entropy-based reliability for configurable and scalable distributed storage systems," Mathematics, vol. 10, no. 1, p. 84, 2021, doi: 10.3390/math10010084.

[3] C. Gentry, A fully homomorphic encryption scheme. PhD Thesis: Stanford University, 2009.

[4] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in Advances in Cryptology – CRYPTO 2013, 2013, pp. 75–92, doi: 10.1007/978-3-642-40041-4_5.

[5] A. Tchernykh, M. Babenko, E. Shiriaev, B. Pulido-Gaytan, J. M. Cortés-Mendoza, A. Avetisyan, A. Yu Drozdov, and V. Kuchukov, "An efficient method for comparing numbers and determining the sign of a number in RNS for even ranges," Computation, vol. 10, no. 2, p. 17, 2022, doi: 10.3390/computation10020017.

[6] B. Pulido-Gaytan, A. Tchernykh, J. M. Cortés-Mendoza, M. Babenko, G. Radchenko, A. Avetisyan, and A. Yu Drozdov, "Privacy-preserving neural networks with homomorphic encryption: Challenges and opportunities," Peer Peer Netw Appl, vol. 14, no. 3, pp. 1666–1691, 2021, doi: 10.1007/s12083-021-01076-8.

[7] W. Jung, E. Lee, S. Kim, J. Kim, N. Kim, and K. Lee, "Accelerating fully homomorphic encryption through architecture-centric analysis and optimization," IEEE Access, vol. 9, pp. 98772–98789, 2021, doi: 10.1109/ACCESS.2021.3096189.

[8] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in International Conference on the Theory and Application of Cryptology and Information Security, 2017, pp. 409–437, doi: 10.1007/978-3-319-70694-8_15.

[9] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," 2019, pp. 347–368. doi: 10.1007/978-3-030-10970-7_16.

[10] B. Pulido-Gaytan, A. Tchernykh, F. Leprévost, P. Bouvry, and A. Goldman, "Toward understanding efficient privacy-preserving homomorphic comparison," IEEE Access, vol. 11, pp. 102189–102206, 2023, doi: 10.1109/ACCESS.2023.3315655.

[11] B. Pulido-Gaytan and A. Tchernykh, "Self-learning activation functions to increase accuracy of privacy-preserving convolutional neural networks with homomorphic encryption," PLoS ONE 19(7), p. e0306420, doi: 10.1371/journal.pone.0306420.

[12] M. Babenko, A. Tchernykh, B. Pulido-Gaytan, A. Avetisyan, S. Nesmachnow, X. Wang, and F. Granelli, "Towards the sign function best approximation for secure outsourced computations and control," Mathematics, vol. 10, no. 12, p. 2006, 2022, doi: 10.3390/math10122006.

[13] F. Piazza, A. Uncini, and M. Zenobi, "Artificial neural networks with adaptive polynomial activation function," in Proc. of the IJCNN, 1992, pp. 343–349.

[14] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," Adv Neural Inf Process Syst, p. 27, 2014.

[15] A. Gautier, Q. N. Nguyen, and M. Hein, “Globally optimal training of generalized polynomial neural networks with nonlinear spectral methods,” Adv Neural Inf Process Syst, p. 29, 2016.

[16] N. G. Timmons and A. Rice, “Approximating activation functions,” arXiv preprint:2001.06370, 2020.

[17] M. Goyal, R. Goyal, and B. Lall, “Learning activation functions: A new paradigm for understanding neural networks,” arXiv:1906.09529, 2019.

[18] R. T. Gregory and E. V. Krishnamurthy, Methods and applications of error-free computation. Springer Science & Business Media, 2012.

[19] J. H. Cheon, D. Kim, and D. Kim, “Efficient homomorphic comparison methods with optimal complexity,” in Advances in Cryptology – ASIACRYPT 2020, 2020, pp. 221–256, doi: 10.1007/978-3-030-64834-3_8.

[20] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy,” in 33rd International Conference on Machine Learning, 2016.

[21] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, “Improved security for a ring-based fully homomorphic encryption scheme,” in International Conference on Cryptography, 2013, pp. 45–64, doi: 10.1007/978-3-642-45239-0-4.

[22] M. Albrecht, S. Bai, and L. Ducas, “A Subfield Lattice Attack on Overstretched NTRU Assumptions,” in Advances in Cryptology – CRYPTO 2016, 2016, pp. 153–178, doi: 10.1007/978-3-662-53018-4_6.

[23] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, “Privacy-preserving classification on deep neural network,” IACR Cryptol. ePrint Arch, vol. 35, 2017.

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

[25] Zvika Brakerski, “Fully homomorphic encryption without modulus switching from classical GapSVP,” in Advances in Cryptology – CRYPTO 2012, 2012, pp. 868–886, doi: 10.1007/978-3-642-32009-5_50.

[26] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, “Fast homomorphic evaluation of deep discretized neural networks,” in Advances in Cryptology – CRYPTO 2018, 2018, pp. 483–512, doi: 10.1007/978-3-319-96878-0_17.

[27] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade, “TAPAS: Tricks to accelerate (encrypted) prediction as a service,” in 35th International Conference on Machine Learning, 2018, pp. 4490-4499.

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

[29] E. Hesamifard, H. Takabi, and M. Ghasemi, “CryptoDL: Deep neural networks over encrypted data,” arXiv:1711.05189. 2017.

[30] Z. Liao, J. Luo, W. Gao, Y. Zhang, and W. Zhang, “Homomorphic CNN for privacy preserving learning on encrypted sensor data,” in 2019 Chinese Automation Congress (CAC), 2019, pp. 5593–5598. doi: 10.1109/CAC48633.2019.8996767.

[31] A. Brutzkus, O. Elisha, and R. Gilad-Bachrach, “Low latency privacy preserving inference,” in 36th International Conference on Machine Learning, 2019, pp. 1295–1304.

[32] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, “NGraph-HE: A graph compiler for deep learning on homomorphically encrypted data,” in Proceedings of the 16th ACM International Conference on Computing Frontiers, 2019, pp. 3–13, doi: 10.1145/3310273.3323047.

[33] X. Jiang, M. Kim, K. Lauter, and Y. Song, “Secure outsourced matrix computation and application to neural networks,” in Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1209–1222. doi: 10.1145/3243734.3243837.

[34] A. Falcetta and M. Roveri, “Privacy-preserving deep learning with homomorphic encryption: An introduction,” IEEE Comput Intell Mag, vol. 17, no. 3, pp. 14–25, 2022, doi: 10.1109/MCI.2022.3180883.

[35] 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. Top. Comput., vol. 9, no. 3, pp. 1330-1343, 2020, doi: 10.1109/TETC.2020.3014636.

[36] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, J.-S. No, “Privacy-preserving machine learning with fully homomorphic encryption for deep neural network,” IEEE Access, vol. 10, pp. 30039–30054, 2022, doi: 10.1109/ACCESS.2022.3159694.

[37] HomomorphicEncryption.org, Homomorphic encryption security standard. Toronto, Canada, 2018. https://homomorphicencryption.org/standard

[38] Y. LeCun and C. Cortes, MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[39] J. Lee, E. Lee, J.-W. Lee, Y. Kim, Y.-S. Kim, and J.-S. No, “Precise approximation of convolutional neural networks for homomorphically encrypted data,” IEEE Access, vol 11, pp. 62062-62076, 2023, doi: 10.1109/ACCESS.2023.3287564.

[40] L. N. Smith and N. Topin, “Super-convergence: Very fast training of neural networks using large learning rates,” arXiv:1708.07120, 2017.

[41] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: surpassing human-level performance on Imagenet classification,” in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034.

[42] J. M. Cortes-Mendoza, G. Radchenko, A. Tchernykh, B. Pulido-Gaytan, M. Babenko, A. Avetisyan, P. Bouvry, A. Zomaya, “LR-GD-RNS: Enhanced privacy-preserving logistic regression algorithms for secure deployment in untrusted environments,” in 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 770–775, doi: 10.1109/CCGrid51090.2021.00093.