Activate Me!: Designing Efficient Activation Functions for Privacy-Preserving Machine Learning with Fully Homomorphic Encryption

Nges Brian Njungle and Michel A. Kinsy
Secure, Trusted, and Assured Microelectronics (STAM) Center
Ira A. Fulton Schools of Engineering
Arizona State University, Tempe, AZ 85281, USA
Emails: nnjungle@asu.edu, mkinsy@asu.edu

***Abstract*— The rapid integration of machine learning into applications in sensitive domains like healthcare and defense raises serious privacy and security concerns. These applications require strong privacy protections, as they rely on large amounts of sensitive data for both training and inference. Fully Homomorphic Encryption (FHE) offers a promising solution by allowing computations on encrypted data, thereby preserving confidentiality throughout the entire machine-learning pipeline. However, FHE only supports linear operations natively. This poses challenges for implementing non-linear activation functions, which are crucial for modern machine learning applications, under FHE constraints.**

In this study, we design, implement, and evaluate activation functions optimized for FHE-based machine learning applications. We focus on two widely used functions: the Square function and the Rectified Linear Unit (ReLU). Our experiments utilize the LeNet-5 and ResNet-20 architectures implemented with the CKKS scheme from the OpenFHE library.

For ReLU, we compare two approaches. First, we explore the popular low-degree polynomial approximation approach. Second, we introduce a novel scheme-switching technique that also securely evaluates ReLU under FHE constraints. Our results show that the square function is highly effective in shallow models like LeNet-5, achieving 99.4% accuracy with an inference time of 128 seconds per image. In deeper networks like ResNet-20, ReLU is more appropriate. The FHE-based ResNet-20 model implemented with ReLU polynomial approximation resulted in an accuracy of 83.8% and an inference time of 1,145 seconds per image. Further, our proposed scheme-switching algorithm achieved a higher accuracy of 89.8%, with an increased inference time of 1,697 seconds per image on a ResNet-20 model as well. These results highlight the key trade-off to be considered when selecting activation functions for FHE-based machine learning applications. The activation functions that reduce computation time tend to significantly lower accuracy, while those that preserve accuracy incur higher computational costs.

I. Introduction

Machine learning (ML) has emerged as a cornerstone of Artificial Intelligence (AI) and is often regarded a key driver for the Fourth Industrial Revolution [1], [2]. Over the past decade, it has experienced unprecedented growth, becoming an essential asset to technology across various sectors, including healthcare, finance, and defense. The ability of ML to derive insights from vast datasets has proven invaluable in critical systems such as medical diagnostics, predictive healthcare, and military decision making [2]. In defense, for example, ML is used in high-stakes applications such as autonomous weapon systems, threat detection, cybersecurity, and intelligence analysis [3]. These capabilities enable faster, data-driven decision making while significantly enhancing operational efficiency and mission effectiveness. However, the successful deployment of ML in these critical applications is highly dependent on access to large volumes of sensitive data for both training and inference. This dependency also presents serious privacy and security challenges. The privacy and security of data are particularly acute in military applications, since the data involved are often classified or highly confidential. Unauthorized access, leakage, or misuse of these data could compromise national security, operational plans, or strategic assets. As a result, ensuring the security and privacy of data in ML workflows in military systems is not only a technical necessity but a matter of national security [4].

Traditional cryptographic methods, although effective at securing data during storage or transmission, fall short when it comes to protecting data during processing. In most cases, data must be decrypted before it can be processed, potentially exposing it to unauthorized access [5]. Fully Homomorphic Encryption (FHE) has emerged as a promising solution to this drawback by enabling computations directly on encrypted data [6]. It ensures that privacy and security are maintained throughout the entire life cycle of data, making it particularly valuable for privacy-preserving machine learning (PPML) applications, where data security and privacy are paramount. Despite its potential, the application of FHE in ML models presents two key challenges: a very high computational cost of evaluating models and the inability to efficiently evaluate non-linear functions [7]. Artificial neural networks are the dominant sub-field of ML today. These networks depend on non-linear activation functions to introduce complexity and enable effective learning. However, the FHE schemes of today typically support only linear operations. This limitation severely restricts the size and practical application of artificial neural networks under FHE constraints [8].

Prior research on the intersection of ML and FHE has centered on enhancing computational efficiency through various optimizations in the design of FHE-based models. In contrast, this work is focused on the design and implementation of activation functions tailored for FHE-based ML models. We examine two commonly used activation functions in FHE-based neural networks architectures: the square function and the Rectified Linear Unit (ReLU). Early FHE-compatible models predominantly employed simple polynomial activations, with the square function being the most widely used. However, recent advancements in the field have shifted FHE-based models toward the adoption of polynomial approximations of the ReLU activation function [9]. While the approximation of ReLU approach resulted in performance improvements, there have been little systematic investigation regarding the contexts and model architectures where different activation functions perform best under FHE constraints. This work bridges that gap by implementing FHE-based neural networks using various activation functions and rigorously evaluating their effectiveness.

We evaluate the performance of these different activation functions using the LeNet-5 and ResNet-20 architectures. These architectures represent shallow and deep artificial neural network models, respectively. For the non-linear ReLU activation function, we implement the widely adopted polynomial approximation technique. We also introduced a novel algorithm that leverages FHE scheme-switching capabilities to enable a high precision evaluation of ReLU under FHE constraints. We compare the approaches based on inference latency and accuracy, emphasizing the trade-offs involved in designing activation functions for FHE-based ML models. Our results indicate that the square function is highly effective for shallow networks like LeNet-5, while ReLU proves more suitable for deeper architectures like ResNet-20. The polynomial approximation-based approach delivered a relatively low inference latency but introduced a higher accuracy degradation. In contrast, our scheme-switching method achieved a higher model accuracy with minimal degradation, albeit at the cost of increased computational overhead. These findings provide important insights into the design of FHE-compatible activation functions and contribute to the development of more practical yet efficient PPML models and solutions. Our main contributions in this work are as follows:

- Design and Implementation of Activation Functions for FHE-based ML models namely, the square and ReLU activation functions.
- Introduction of a novel algorithm using the scheme-switching capabilities of FHE to evaluate the ReLU activation function in the encrypted domain.
- We conduct a comprehensive evaluation of the square function, the polynomial approximation of ReLU, and the proposed scheme-switching ReLU method using LeNet-5 and ResNet-20 models under FHE constraints. We highlight the trade-offs of these approaches in terms of model accuracy and inference latency.

II. Related Works

Artificial neural networks have become the dominant approach in modern machine learning applications. They are generally composed of interconnected linear layers and activation functions. Activation functions play a crucial role in these models as they enable them to capture complex, non-linear patterns in data [10]. In recent years, artificial neural network models incorporating FHE for enhanced security and privacy have made significant advancements. However, research has predominantly focused on enhancing the efficiency of encrypted inference. While notable advancements have been achieved in this direction, training artificial neural networks under FHE constraints remains an open challenge. This is because of the substantial computational overhead and inherent accuracy loss of models when trained in the encrypted domain. The high computational cost and accuracy loss associated with FHE-based training arises from the complexity of operations like backpropagation, which must be executed on large amounts of encrypted data while maintaining high precision. Thus, making the process a very expensive and challenging computational task [11].

PPML applications utilizing FHE are generally categorized into two generations. The first-generation approaches are characterized by limited model depth and high computational overhead. The second-generation methods introduced advanced cryptographic optimizations including approximation of ReLU and Bootstrapping. These advancements significantly improved efficiency and enabling more practical applications of FHE in machine learning applications. There are other existing FHE-based ML applications which leverage other privacy preserving techniques like multi-party computation [12] to facilitate the evaluation of non-linear activation functions with examples being Gazelle [13], MiniONN [14], and XONN [15]. However, these works are generally considered as hybrid due to their integration of multiple techniques to provide privacy and security thus we do not employ this approach in this work. In this work, we focus on strict FHE-based ML applications specifically artificial neural networks.

The first generation of FHE-based neural networks began with CryptoNets from Microsoft Research in 2016. The framework achieved a 99% accuracy on the MNIST dataset using a neural network custom architecture designed specifically for this task [16]. CryptoNets main adjustment to neural network architectures was the use of the square function as their activation function which removed the complexity involved with non-linear activation functions. In 2017, Ehsan et al. proposed CryptoDL, which shitted from the simple square to a more complex low degree polynomial as the activation function [17]. Another notable contribution came from Al Badawi et al. who accelerated HE inference using GPUs with a custom AlexNet-like architecture [18] still utilizing the square activation function. Other works in this generation such as; E2DM [19], and DiNN [20] have also adopted the approach of using simple low-degree polynomial functions as activation functions.

The second generation of FHE-based neural networks introduced more sophisticated architectures with significant performance improvements. These works employed sophisticated techniques like bootstrapping which enables computations of arbitrary depth networks. They also introduced the evaluation of non-linear activation functions using polynomial approximation methods to enable privacy-preserving computation in complex and deep neural networks under FHE constraints. During training, non-linear functions are utilized (specifically the ReLU activation function). For encrypted inference, the non-linear functions are replaced with their polynomial approximations. A primary focus of various works in this generation has been optimizing resource usage and performance in HE-based neural networks. Works in this generation generally adopt the CKKS or TFHE homomorphic encryption schemes. The TFHE scheme is known for its fast and efficient bootstrapping after every multiplication called programmable bootstrapping [21]. This approach enables the evaluation of binary neural networks of arbitrary depth circuits while maintaining small ciphertext sizes with minimal precision loss. The most recent and most efficient of these works is the work of Benamira et al. [22], which proposed a TFHE-based architecture that achieved 74.1% accuracy on the CIFAR-10 dataset using a custom TFHE-CNN friendly architecture. The main limitation of these works stems from the fact that TFHE is inherently slow in large-scale data processing compared to other FHE schemes, like CKKS and BGV. This is due to its lack of support for parallel data processing. Also, the TFHE scheme operates on bit-wise data, which contrasts the floating-point arithmetic generally used in modern machine learning models applications. These limitations make TFHE not very tractable for processing complex or deep neural network architectures [23]. On the other hand, the CKKS scheme has emerged as the most compatible FHE scheme for ML works given that it is the only mainstream FHE scheme that supports floating-point arithmetic and parallel data processing [24] [25]. These capabilities significantly accelerate computations while providing a possible direct mapping of functionalities with unencrypted ML works. In 2022, Lee et al. proposed an FHE inference system that achieved 91.31% accuracy on the CIFAR-10 dataset in 2,271 seconds [25] using the polynomial approximation of the ReLU activation function. Building on this, Kim et al. introduced a more efficient approach, achieving 92.04% accuracy on CIFAR-10 with a single image inference time of 255 seconds [26]. Most recently, Rovida et al. proposed an optimized implementation of Kim et al.'s work, achieving 91.53% accuracy on the CIFAR-10 dataset in 260 seconds while using only 15.1 GB of memory [27]. All works mentioned have used the polynomial approximation of the ReLU activation function to evaluate ResNet-20 models.

Unlike most FHE-based ML works today, which primarily focus on optimizing FHE inference while applying the approximation of ReLU or using low-degree polynomials like the square function, our work takes a different approach. We focus on evaluating different activation functions suitable for PPML solutions under FHE constraints. By analyzing different methods, we identify where the different activation functions are most effective in constructing FHE-based models. Through a comprehensive analysis of results, this work provides insights that will assist in selecting activation functions that best fit the privacy preserving needs of different applications.

III. Background

A. Fully Homomorphic Encryption

Homomorphic Encryption (HE) is a cryptographic paradigm that enables direct computations on encrypted data without requiring decryption. This concept was originally introduced in 1978 under the term *Privacy Homomorphism* [28], shortly after the introduction of the RSA public key encryption scheme [6]. The introduction of HE marked a significant advancement in secure computation, allowing encrypted data to be manipulated while preserving confidentiality. A major breakthrough in this sub field of cryptography occurred in 2009 when Craig Gentry introduced the first Fully Homomorphic Encryption (FHE) scheme [29] in his PhD thesis. FHE enables an arbitrary number of computations on encrypted data, overcoming the limitations from earlier HE schemes that only supported a restricted set of operations and only very limited depth circuits. Gentry's work laid the foundation for practical FHE by introducing two fundamental techniques: *bootstrapping* and *squashing*. These innovations provided a structured approach to extending somewhat homomorphic encryption schemes to fully homomorphic ones. Gentry's scheme was based on ideal lattices, that inherently support homomorphic addition and multiplication for circuits of very limited depth. A critical challenge and limitation of lattice based HE schemes is noise accumulation within ciphertexts. As computations proceed, the noise level increases, and if it surpasses a certain threshold, the ciphertext becomes undecipherable thus rendering the encrypted message to be irretrievable. To address this issue, Gentry used an expensive computational operation called bootstrapping. He defined bootstrapping as a mechanism that periodically reduces noise within a ciphertext, thereby enabling indefinite computations on encrypted data. This approach established a foundation for subsequent improvements in FHE schemes, making them increasingly efficient and practical for real-world applications.

While other mainstream FHE schemes such as Brakerski-Gentry-Vaikuntanathan (BGV) [30], Brakerski/Fan-Vercauteren (BFV) [31], and Fast Fully Homomorphic Encryption over the Torus (TFHE) [21] exist, the Cheon-Kim-Kim-Song (CKKS) scheme [24] is the most suitable for machine learning applications. It is the only mainstream FHE scheme that supports both approximate arithmetic and parallel data processing, therefore being the only scheme that aligns perfectly well with the needs of modern machine learning workflows and workloads. Figure 1 illustrates a privacy preserving outsourced ML inference scenario utilizing HE. In this setup, a user is seeking to perform inference on an encrypted model stored in the cloud. First, they encrypt their input data using their public key and transmit it to the cloud. The cloud server processes the encrypted data using the encrypted model without ever decrypting it. Once the processing is completed, it returns the encrypted inference result to the user. Upon receiving the encrypted result, the user then decrypts it using their private key to gain access to the model's inference output. It is important to note that, at no point in this process did the cloud provider or anyone else gain access to the input data or inference result except for the user. Also, the cloud provider cannot extract any information from the encrypted model, since they do not have access to the private key. Furthermore, even in the event of a cloud breach, the user data and model stay protected, since all information is maintained in its encrypted form.

This mechanism upholds confidentiality of both the user's data and the model, making HE a robust solution for secure cloud-based machine learning inference.

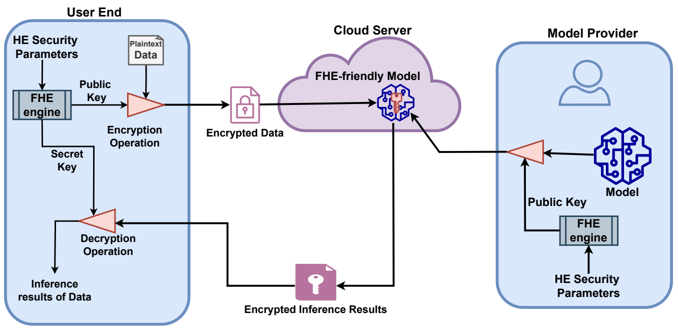

Fig. 1: Privacy-preserving machine learning inference in an outsourced cloud scenario showing a user inferring encrypted data over an encrypted model.

B. Cheon-Kim-Kim-Song (CKKS) Scheme

In 2016, Cheon, Kim, Kim, and Song introduced the CKKS scheme, a HE scheme specifically designed for approximate arithmetic [24]. Its security is based on the Ring Learning with Errors (RLWE) problem, a structured extension of the Learning with Errors (LWE) problem proposed by Regev in 2005 [32] and later refined for polynomial rings by Lyubashevsky et al. [33]. The LWE problem involves recovering a hidden vector from noisy linear equations generated using random samples. Formally, given a matrix $A \in \mathbb{Z}_q^{m\times n}$ and a vector $b \in \mathbb{Z}_q^m$, the goal is to find an unknown vector $s \in \mathbb{Z}_q^n$ satisfying:

$$As + e = b \bmod q, \tag{1}$$

where e is a noise vector sampled from a predefined error distribution, and q is a large prime modulus.

Expanding on this, the RLWE problem generalizes LWE to polynomial rings, leveraging their algebraic properties for computational efficiency. A key advantage of CKKS is its ability to encode multiple plaintext values into a single polynomial through a technique known as "slot packing". This feature enables efficient parallel computations using the Single Instruction Multiple Data (SIMD) paradigm, significantly accelerating homomorphic operations through batch processing of data. The CKKS scheme defines a range of homomorphic operations, including addition, subtraction, multiplication, rotation, and bootstrapping. These operations enable users to build a wide range of applications that support encrypted data processing with CKKS making it well-suited for PPML and other critical applications that require privacy computations on real-valued data.

C. Machine Learning

Machine learning (ML) is a sub-field of artificial intelligence that empowers computers to recognize patterns within data and generate predictions or decisions without requiring explicit programming [34]. The field of ML is broadly categorized into three paradigms: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training models using labeled datasets, enabling them to make precise predictions based on known input-output pairs [35]. In contrast, unsupervised learning identifies hidden structures and patterns within unlabeled data, uncovering insights without predefined outcomes [36]. Reinforcement learning is distinct from these approaches. It employs a trial-and-error mechanism where models interact with an environment, then develop optimal strategies through rewards and penalties [37].

A significant advancement within the field of ML is the area of deep learning. Deep learning utilizes artificial neural networks (mainly convolution neural networks) with multiple layers to process complex and high-dimensional data representations [38]. These networks typically comprise of linear layers and activation functions. The linear layers generally include the convolution layers, the pooling layers, and the fully connected layers. These linear layers have been widely studied under FHE constraints as they generally account for the bulk of the computational cost of FHE-based models. For this reason, we will not discuss them in this work. Instead, we will focus on activation functions since they have not been well studied under FHE constraints. Activation functions are particularly interesting as they introduce nonlinearity in learning, allowing ML model to capture intricate relationships within data. Commonly used activation functions in modern ML applications include the Rectified Linear Unit (ReLU), Sigmoid, and Hyperbolic Tangent (Tanh), each contributing uniquely to model efficiency and performance. Deep learning techniques can be applied across all three ML paradigms, enhancing supervised learning with highly accurate models, enabling advanced clustering in unsupervised learning, and improving decision-making capabilities in reinforcement learning.

IV. FHE-based Activation Functions

Activation functions play a fundamental role in ML models. Particularly in artificial neural networks, they introduce nonlinearity and enabling complex pattern learning [39]. This non-linearity allows networks to learn complex patterns and relationships in data that linear models cannot capture. By determining how neurons process and transmit signals through layers, activation functions fundamentally shape a network's ability to learn and generalize from data. Traditional activation functions like the sigmoid and hyperbolic tangent (tanh) were widely used in early neural networks due to their smooth, differentiable properties. However, they often suffered from vanishing gradient issues, which made training deep networks difficult [40], [41]. In response, more recent advancements have favored the Rectified Linear Unit (ReLU) and its variants because they provide faster convergence, reduce the gradient-related problems, and encourage sparse activations [42], [43].

While activation functions are relatively straightforward to implement in the plaintext domain, they present significant challenges in PPML settings, particularly when using FHE. For instance, ReLU, is non-polynomial and non-smooth, making it incompatible with the algebraic operations supported by most FHE schemes. To address this issue, researchers have proposed FHE-based alternatives based on low-degree polynomial functions including simple square functions and polynomial approximations of ReLU. In this section, we examined the design and implementation of these activation functions in the FHE context, highlighting their algorithms, benefits, and the trade-offs they entail.

A. *Square Activation function*

Popular activation functions used in ML, such as the ReLU and sigmoid functions, involve non-polynomial operations that are difficult to compute under FHE constraints. In contrast, the square function is inherently polynomial, making it well-suited for homomorphic evaluation. Its algebraic simplicity aligns naturally with the linear structures supported by most FHE schemes, making it both easy to understand and straightforward to implement. Due to these advantages, the square activation function gained significant attention in the context of FHE-based PPML solutions. As a low-degree polynomial, it was the first activation function adopted for encrypted neural networks computation. Among other low-degree polynomials, it is valued for its simplicity, computational efficiency, and compatibility with homomorphic operations. Mathematically, the square activation function is defined in Equation 2.

$$f(x) = x^2 \tag{2}$$

where x is the input to the function f.

The implementation of the square activation function using the CKKS scheme is outlined in Algorithm 1.

Algorithm 1 The Square Function Algorithm in FHE

```
1: **Input:** $x$ (input ciphertext)
2: **Output:** $y$
3: $y \leftarrow \mathit{CKKSmultiply}(x, x)$
4: **return** $y$
```

The square activation function is smooth and differentiable everywhere, which facilitates stable gradient propagation during training. However, its derivative, given by $f'(x) = 2x$, is also a linear function. Unlike non-linear activation functions whose gradients exhibit non-linear behavior promoting piecewise non-linearity and sparsity, the square function's linear derivative introduces certain optimization challenges. In particular, the lack of saturation regions and the absence of a thresholding mechanism reduce the network's capacity to capture complex hierarchical features effectively. Another key limitation of the square function is its inherent non-negativity. By squaring all inputs, it eliminates sign information and produces only non-negative outputs without any accumulations. This behavior can lead to redundancy in activation patterns and increase the correlation among neuron outputs, which may negatively affect the model's ability to generalize. Moreover, the unbounded nature of the derivative $2x$ for large input magnitudes can cause gradient instability during training. This is especially problematic in deep architectures, where the accumulation of large gradients can result in exploding gradient phenomena, ultimately destabilizing the learning process.

These challenges make it difficult to train deep neural networks effectively using the square activation function. Modern architectures like ResNet-20, rely on residual connections that perform best when activation functions maintain controlled gradient norms. The unbounded growth of the square function's derivative disrupts this balance, leading to unstable weight updates. As a result, while the square function is computationally efficient and works well in shallow FHE-based models, it is not suitable for deeper architectures. For instance, training the ResNet-20 model with the square activation function was proved infeasible due to instability during backpropagation. The exploding gradients prevented effective learning and completely hindered the network's ability to generalize.

B. *ReLU Activation Function*

Rectified Linear Unit (ReLU) is the most widely adopted activation function in ML today due to its simplicity and effectiveness [43]. ReLU applies a threshold operation where all negative values are set to 0. This non-linearity accelerates convergence and helps reduce the vanishing gradient problem in training. Equation 3 shows the mathematical representation of the ReLU activation function.

$$f(x) = max(0, x) \tag{3}$$

Unlike traditional activation functions such as sigmoid and tanh, that suffer from vanishing gradient problems, ReLU preserves gradient flow for positive inputs, allowing for faster convergence and improved performance in deep architectures. By setting all negative inputs to zero, ReLU introduces sparsity in activations, reducing computational complexity and enhancing model generalization by mitigating unnecessary neuron activations. This sparsity also contributes to an implicit form of regularization, preventing over-fitting in large networks.

1) Polynomial Approximation of ReLU: In FHE-based ML models, the implementation of ReLU poses challenges due to its non-polynomial nature, making direct computation on encrypted data inefficient especially in the CKKS scheme. Previous FHE-based neural networks implementations have focused on Polynomial approximation of the ReLU activation function mainly using the Chebyshev's Polynomial Approximation.

Chebyshev polynomials form a family of orthogonal polynomials defined on the interval $[-1, 1]$. They are highly effective in approximating complex, non-linear functions with both efficiency and precision [44]. These polynomials are denoted by $T_n(x)$ and are defined recursively as follows:

$$T_0(x) = 1, \tag{4}$$
$$T_1(x) = x, \tag{5}$$
$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad \text{for } n \geq 2. \tag{6}$$

A distinctive feature of Chebyshev polynomials is the distribution of their roots. A polynomial of degree n has exactly n roots, all of which are located within the interval $[-1, 1]$. These roots are expressed in the following form:

$$x_k = \cos\left(\frac{k\pi}{n}\right), \quad \text{for } 0 \leq k < n. \tag{7}$$

The Chebyshev polynomials offer an efficient way to express non-linear functions as a sum of polynomials. This approach proves especially useful in FHE applications as polynomials seamlessly integrate with the underlining data representations. By employing Chebyshev polynomial approximations, the ReLU function can be approximated securely within the encrypted domain.

To correctly apply Chebyshev approximations, it is important to normalize the input data to lie within the interval $[-1, 1]$. We achieved this by introducing a scaling factor β that adjusts the input vector accordingly. The input vector is scaled by multiplying it with $\frac{1}{\beta}$, as shown in Equation 8:

$$x_{\mathrm{s}} = \frac{1}{\beta} \cdot x \tag{8}$$

The parameter β is selected through an analysis of values across different datasets and networks. This process ensures that the data is accurately normalized while maintaining the representational integrity of the ReLU activation function under FHE constraints. When $\beta > 1$, a scaling mask is applied to the input vector $\mathbf{x}$ to ensure proper scaling of data before processing. Once scaled, the modified input is passed to the Chebyshev approximation function. The results are then scaled back by multiplying them by β, as shown in Algorithm 2.

Algorithm 2 Secure ReLU Using Homomorphic Encryption

```
1: Input: $x$, $\beta$
2: Output: $y$
3: Initialization: $D$, $v \leftarrow x$
4: if $\beta > 1$ then
5:     $v \leftarrow$ CKKSMultiply$(x, \beta)$
6: end if
7: Define $f(z) : f(z) \leftarrow 0$ if $z < 0$; $f(z) \leftarrow \beta \cdot z$ otherwise
8: $y \leftarrow$ ChebyshevFunction$(f(z), v, D)$
9: return $y$
```

In Algorithm 2, D is the degree of the Chebyshev polynomial used for approximating the function. The value of D determines the accuracy of the Chebyshev Approximation of the ReLU function. The higher the degree, the better the approximation but the higher the computational cost. To determine the right value of D to use, we conducted an experiment with varying values for the polynomial degree between 10 and 100. We studied how well the function approximated the results of the ReLU function. We determined that when D is equal 50, we obtain a good balance between accuracy and the resources used for the polynomial approximation of the ReLU activation function. This value of D consumes a noise budget equivalent to that of eight multiplications thus efficient for use in applications with FHE parameters that can lead to a multiplication depth after bootstrapping to still be greater than eight.

C. Secure ReLU Evaluation via Scheme Switching

FHE schemes exhibit varying degrees of efficiency and computational capabilities, making them suitable for different types of encrypted computations and operations. The CKKS scheme, in particular, is well-suited for approximate arithmetic and supports efficient parallel computation. These properties make CKKS the optimal choice for evaluating linear operations in FHE-based ML applications. However, CKKS is inherently limited in its ability to perform non-linear operations, such as the comparison operation required for thresholding in the ReLU activation function.

In contrast, the FHEW [45] and TFHE schemes [21] are specifically designed for efficient evaluation of boolean gates. They enable precise computation of non-linear functions such as comparisons, sign determination, and the floor function [21]. These functions are expressed as boolean circuits which are then constructed from basic logic gates such as AND, OR, NOR, and XOR defined in these schemes. This gate-based approach allows for accurate discrete computation of functions under FHE constraints. While FHEW and TFHE offer high accuracy in such boolean computations, they do not support SIMD-style parallelism. Consequently, they introduce significant computational overhead when applied to large-scale data processing tasks like those commonly encountered in ML applications.

To leverage the strengths of both encryption schemes, we adopt a scheme-switching approach that integrates CKKS with TFHE. Specifically, CKKS is used to efficiently evaluate all linear layers in the model, while TFHE is employed in the evaluation of non-linear activation functions. During inference, ciphertexts are switched back and forth between the two schemes depending on the type of layer and operation been evaluated. This hybrid strategy enables efficient computation while maintaining compatibility with the constraints of FHE and capitalizing on the complementary strengths of both schemes. The concept of scheme switching between different homomorphic encryption schemes was first introduced by Christina et al. in 2018 under the name CHIMERA [46]. This technique is feasible between CKKS and TFHE because both schemes are rooted in the LWE problem. As a result, ciphertexts in either scheme share the same foundational algebraic structures, making translation between schemes mathematically viable. In essence, scheme switching involves transforming the underlying representation of ciphertexts from one scheme to another while preserving the encrypted data and its semantic integrity.

In this method of evaluating the ReLU activation function, we first determine the sign of each encrypted value within a CKKS SIMD ciphertext by switching to the TFHE scheme, which supports precise comparison operations. Once the signs are computed in TFHE, we switch back to CKKS and use the resulting ciphertext to construct a binary mask. This mask is then homomorphically multiplied with the original CKKS ciphertext, effectively zeroing out the negative values and thereby implementing a more precise, secure, and privacy-preserving ReLU activation function. This algorithm consumes just a single multiplication depth in the noise budget of ciphertext. The complete procedure is outlined in Algorithm 3.

This proposed scheme-switching technique effectively integrates the strengths of CKKS and TFHE, enabling efficient and secure ReLU activation in encrypted neural networks. CKKS provides high throughput in evaluating linear layers, while TFHE assist in a precise computation of the comparison operation needed to effectively evaluate the non-linear ReLU activation function. This hybrid approach allows for the construction of high-precision PPML models under FHE constraints. Although the scheme-switching process of ReLU still results in some errors, the resulting errors are much smaller than those incur in the approximation when using low-degree polynomials.

Algorithm 3 Secure Scheme-Switching ReLU Algorithm

```
1: **Input:** $c_{\text{enc}}$ (encrypted input vector), $c_{\text{sec}}$ (second ciphertext), vector_size
2: **Output:** $c_{\text{result}}$
3: Set slots of $c_{\text{enc}}$ to vector_size
4: $tc_{\text{enc}} \leftarrow \text{CKKSSwitchToTFHEW}(c_{\text{enc}})$
5: $tc_{\text{sec}} \leftarrow \text{CKKSSwitchToTFHEW}(c_{\text{sec}})$
6: $c_{\text{comp}} \leftarrow \text{TFHECompare}(tc_{\text{enc}}, tc_{\text{sec}}, \text{vector\_size})$
7: $\text{mask} \leftarrow \text{GenerateCKKSPlaintextMask}(1, \text{vector\_size})$
8: $c_{\text{sign}} \leftarrow \text{CKKSSubtract}(\text{mask}, c_{\text{comp}})$
9: $c_{\text{result}} \leftarrow \text{CKKSMultiply}(c_{\text{sign}}, c_{\text{enc}})$
10: **return** $c_{\text{result}}$
```

V. Experiment and Results

All experiments in this study were performed on a AMD Ryzen 9 5900X 12-core processor with 64GB of RAM. We build all our models on OpenFHE v1.2.3 which is the most stable version of the library at the time of development. We chose OpenFHE because it is the most advanced FHE library available to the development community. Additionally, it is the only FHE library that supports implementations for both CKKS and FHEW/TFHE gates [47] [48].

A. Network Architectures

To comprehensively evaluate the performance of these different activation function implementations under FHE, we evaluate two distinct neural network architectures: LeNet-5 and ResNet-20. These models differ significantly in complexity and application domains, making them ideal candidates for assessing the trade-offs between efficiency and accuracy in privacy-centric applications.

LeNet-5 is a relatively simple and shallow artificial neural network originally designed for grayscale image classification tasks such as handwritten digits recognition [49]. Despite its simplicity, LeNet-5 remains one of the most widely studied architectures in ML literature, making it an excellent baseline for evaluating activation functions in FHE settings. For this study, we trained two versions of LeNet-5 on the MNIST dataset using the ReLU activation function and using the square activation function. The training process was conducted in PyTorch, and the learned model weights were exported as CSV files which were then loaded into the encrypted models for inference. Within the encrypted domain, we implemented three different evaluation models. Our first model used the square activation function with equivalent weights exported from the model trained on the square activation function. Our second model used the polynomial approximation of ReLU activation function while our third model used the scheme-switching evaluation of ReLU activation function. The last two FHE-based LeNet-5 models used the weights from the model trained with the ReLU activation functions. Each encrypted model was evaluated using 750 images from the MNIST validation set. Figures 2 illustrate the detailed architectures of the LeNet-5 models used in this study including the input channels, output channels, kernel, padding, and striding configurations.

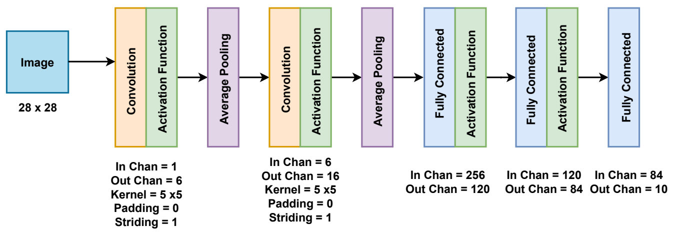

Fig. 2: The LeNet-5 Architecture used in this work:
In Chan = input channels, Out Chan = output channels

The ResNet models are some of the most widely adopted deep learning models for complex image classification tasks today [50]. The core innovation in ResNets is the introduction of residual connections, which mitigate the vanishing gradient problem and enable efficient training of deeper networks. Due to its balance of depth and computational feasibility, ResNet-20 has been commonly employed in prior FHE research and serves as an ideal benchmark for our study.

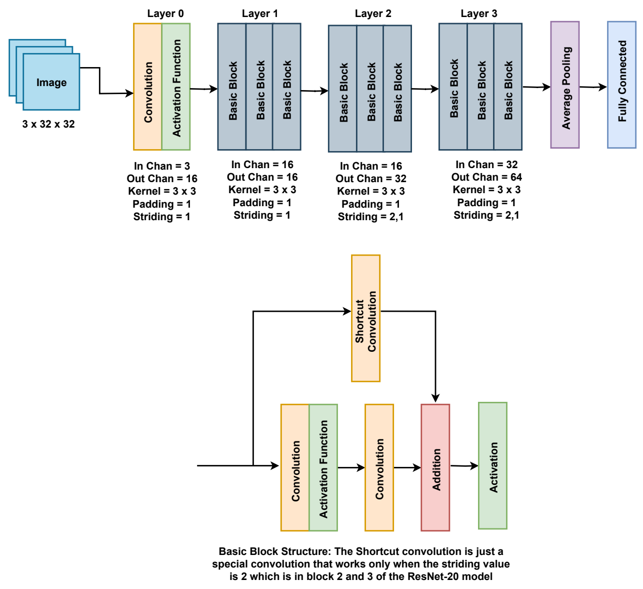

Fig. 3: ResNet-20 Architecture. The Blocks show the ResNet blocks. The Block Structure shows the details and how the convolutions are connected

For our evaluation, we trained a ResNet-20 model on the CIFAR-10 dataset using the ReLU activation function. In contrast to LeNet-5, we found that training ResNet-20 with the square activation function was infeasible due to its limited capacity to approximate non-linear behaviors in deep architectures (see Section IV). This result highlights a fundamental limitation of low-degree polynomial activation functions in privacy-preserving deep learning, as their expressive power diminishes with increasing network depth.

After training the ResNet-20 model with ReLU activation functions, the resulting model weights were also exported as CSV files for use with the FHE-based models to inference on encrypted data. Within the encrypted domain, we implemented two evaluation models: one utilizing a polynomial approximation of ReLU and the other employing the scheme-switching approach introduced in Algorithm 3. We conducted encrypted inference on both models on a subset of 250 images from the CIFAR-10 validation set. The key metrics evaluated were accuracy retention and inference latency for the two ReLU implementation approaches. To achieve fast inference for accuracy evaluation, we utilized the SOL supercomputer at Arizona State University [51]. Latency measurements, however, were collected directly from a local consumer-grade setup using an AMD Ryzen 9 5900X CPU. Figures 3 illustrate the ResNet-20 architectures used in this study including the input channels, output channels, kernel, padding and striding configurations. Its residual connections show the additional convolutions evaluated in the down-sampling stage of ResNets.

Furthermore, since noise accumulation is a critical challenge in FHE, we incorporated bootstrapping operations at necessary points to preserve computational integrity. Table I summarizes the number of bootstrapping operations required for each model to ensure accurate and efficient encrypted inference. The placement of bootstraps within the models were carefully determined based on the estimated noise levels at various stages of the inference process. These estimates were derived from a detailed analysis of ciphertext modulus consumption throughout the computation pipeline. By strategically applying bootstrapping only when the noise threatened to exceed the decryption threshold, we minimized overhead while maintaining correctness in the encrypted domain.

TABLE I: Number of Bootstrapping Operations required in Encrypted Models

Architecture	**Number of Bootstraps used**
LeNet-5 With Square	0
LeNet-5 with Approximation	4
LeNet-5 with Scheme Switch	0
ResNet-20 with Approximation	18
ResNet-20 with Scheme Switching	8

Table II provides an overview of the security parameters used in our encrypted inference experiments. These parameters were carefully chosen to maintain a security level around 128 bits of security with maximum throughput within the models, ensuring robustness against cryptographic attacks while optimizing computational efficiency.

TABLE II: The CKKS Parameter sets used to evaluate the LeNet-5 and ResNet-20

Parameter	**LeNet-5**	**ResNet-20**
Polynomial Degree	16384	32768
Number of Slots	8192	16384
Multiplications after Bootstrap	10	10

B. Results

Table III presents the accuracy of the evaluated neural network models under both plaintext and under FHE constraints, as well as the latency measurements in the encrypted domain. The primary objective of this evaluation was to investigate the trade-offs between computational overhead and model accuracy associated with different methods for evaluating activation functions under FHE constraints. Our analysis compares the three activation function evaluation approaches in the encrypted domain: the low-degree polynomial square function, polynomial approximations of the ReLU function, and the scheme-switching ReLU evaluation method. The results show the inherent trade-offs between accuracy retention and computational efficiency in FHE-based PPML solutions.

For the LeNet-5 architecture, all three activation function methods achieve comparable performance, exhibiting minimal accuracy loss between plaintext and encrypted inference. Notably, the square activation function achieves the highest encrypted accuracy of 99.4%, with only a 0.2% degradation from its plaintext counterpart. However, it offers latency performance higher than that of the ReLU polynomial approximations which significantly outperforms the scheme-switching method in terms of runtime. A key advantage of the square function lies in its seamless compatibility with the CKKS scheme, which allows it to be implemented using a single homomorphic multiplication thereby eliminating the need for bootstrapping and further improving computational efficiency in shallow architectures like LeNet-5.

The ReLU approximation method shows a modest accuracy degradation of 0.3% in the encrypted LeNet-5 model while maintaining the best latency across all evaluations. Meanwhile, the scheme-switching ReLU approach also incurs a 0.3% accuracy loss but introduces significant computational overhead. Specifically, it requires approximately 168 seconds to inference a single image which is about $1.7\times$ the latency of the approximation-based method. This increased cost arises from the additional operations required to switch between CKKS and TFHE schemes, along with the inherently higher computational complexity of TFHE operations. Despite its slower performance, the scheme-switching method still offers latency that is practical for many real-world applications.

In deeper architectures like ResNet-20, the choice of activation function plays a critical role in determining the model's performance. Residual networks, in particular, rely on nonlinear activations to enable effective feature transformation across layers. Consequently, the selection of an appropriate activation function directly affects the model's expressiveness and accuracy. When employing polynomial approximations of the ReLU function within the CKKS framework, we observed an 8.4% accuracy degradation relative to the plaintext model. This substantial drop in performance can be attributed to the limited capacity of low-degree polynomials to accurately approximate the non-linear behavior of ReLU, thereby leading to suboptimal feature transformations during inference. Nevertheless, this method remains computationally efficient, as it avoids the need for scheme-switching or bootstrapping, making it well-suited for latency-sensitive applications where minor accuracy compromises are acceptable. In contrast, the scheme-switching ReLU evaluation method achieves significantly higher accuracy, incurring only a 2.4% drop from the plaintext baseline. This demonstrates the effectiveness of leveraging TFHE's precise operations capabilities to reconstruct the ReLU function more efficiently within the encrypted domain. However, the improved accuracy comes at the cost of increased computational complexity. The scheme-switching model required approximately $1,697$ seconds for inference, representing a $1.5\times$ increase in runtime compared to the polynomial approximation approach.

TABLE III: Accuracy Comparison Between Plaintext and FHE Settings as well as the latency in the FHE setting for the LeNet-5 models and ResNet-20 models

Architecture	**Plaintext (%)**	**Encrypted (%)**	**Latency (s)**
LeNet-5 Square	99.6	99.4	128
LeNet-5 Approximation	99.2	98.9	95
LeNet-5 Scheme Switch	99.2	98.9	168
ResNet-20 Approximation	92.2	83.8	1,145
ResNet Scheme Switch	99.2	89.8	1,697

These results highlight a fundamental trade-off in PPML applications that rely on FHE. While polynomial approximations of ReLU activation functions substantially accelerate encrypted inference, they introduce notable accuracy degradation, particularly in deeper network architectures. Conversely, our introduced scheme-switching methods offer a more faithful representation of ReLU, preserving model accuracy at the cost of increased computational overhead.

With shallow neural networks like LeNet-5, polynomial activation functions, particularly the square function, offer an optimal balance between accuracy and computational efficiency under FHE constraints. This function aligns well with the CKKS encryption scheme and avoids costly bootstrapping operations in this model, making it especially suitable for real-time applications in resource-limited settings. ReLU-based LeNet-5 models also performed well, with the approximation approach achieving the lowest latency in the FHE-based LeNet-5 models. For deeper architectures like ResNet-20, the optimal activation function strategy depends heavily on application-specific requirements. If accuracy preservation is paramount, the scheme-switching ReLU approach provides a robust solution with minimal degradation. However, if inference latency and computational resources are the primary constraints, ReLU approximation methods offer a practical alternative, albeit with a slight compromise in accuracy. Ultimately, these findings underscore that activation function selection in PPML is not a one-size-fits-all decision. Researchers and practitioners must carefully evaluate their deployment requirements in other to balance accuracy, latency, and model depth when selecting an appropriate activation function strategy for FHE-based neural networks.

VI. Conclusion

In this work, we explored the impact of activation functions on the efficiency and accuracy of privacy-preserving neural networks under Fully Homomorphic Encryption (FHE) constraints. We also introduce a novel algorithm for high precision evaluation of the ReLU activation function based on scheme-switching between the TFHE and CKKS schemes. Our results, highlight the trade-offs between the square activation function, ReLU approximation, and our proposed scheme-switching ReLU activation functions in both shallow and deep neural network architectures. For shallow networks like LeNet-5, the square activation function proved to be the most efficient choice, offering the best accuracy with a comparable latency. Additionally, its implementation allowed us to eliminate the need for bootstrapping. This further enhance computational efficiency of models like LeNet-5 in scenarios where bootstrapping implementation is not possible. On the other hand, the challenges that come with low degree polynomial activation functions makes it difficult to adopt the square activation function in deep models. For deeper architectures like ResNet-20, the choice of activation function had a more pronounced effect. The polynomial approximation approach for evaluating ReLU was computationally efficient but resulted in a notable accuracy drop of 8.4%. In contrast, the scheme-switching method preserved accuracy with only a 2.4% degradation. This lower accuracy degradation came at though the cost of increased latency.

These findings highlight a fundamental trade-off in the selection of activation functions for privacy-preserving machine learning application based on FHE. Polynomial activations like the square activation function are computationally efficient and well-suited for shallow models such as LeNet-5, whereas the scheme-switching method is better aligned with deep architectures where maintaining precision is critical. In performance sensitive applications, the ReLU approximation approach emerges as the most practical choice, underscoring the importance of fine-grained tuning to reduce accuracy degradation. In high precision settings, our proposed scheme-switching approach of evaluating the ReLU activation function is the most appropriate though it comes with a higher computational cost compared to the approximation approach. Thus, the choice between activation functions in FHE-based machine learning applications should be guided by the specific accuracy and efficiency requirements of the target application. Future research should prioritize optimizing scheme-switching mechanisms to reduce inference latency without compromising accuracy. Another interesting research direction is the fine-tuning of the polynomial approximation technique to reduce their accuracy degradation rates as well as hybrid applications of these techniques. These advancements will enhance the feasibility and scalability of FHE-based solutions in modern machine learning applications.

References

[1] H. Magd, H. Jonathan, S. A. Khan, and M. El Geddawy, "Artificial intelligence—the driving force of industry 4.0," *A roadmap for enabling industry 4.0 by artificial intelligence*, pp. 1–15, 2022.

[2] D. Virmani, M. A. S. Ghori, N. Tyagi, R. Ambilwade, P. R. Patil, and M. Sharma, "Machine learning: The driving force behind intelligent systems and predictive analytics," in *2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies*. IEEE, 2024, pp. 1–6.

[3] E. J. Alcántara Suárez and V. Monzon Baeza, "Evaluating the role of machine learning in defense applications and industry," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1557–1569, 2023.

[4] M. I. Tariq, N. A. Memon, S. Ahmed, S. Tayyaba, M. T. Mushtaq, N. A. Mian, M. Imran, and M. W. Ashraf, "A review of deep learning security and privacy defensive techniques," *Mobile Information Systems*, vol. 2020, no. 1, p. 6535834, 2020.

[5] R. R. Salavi, M. M. Math, and U. Kulkarni, "A survey of various cryptographic techniques: From traditional cryptography to fully homomorphic encryption," in *Innovations in Computer Science and Engineering: Proceedings of the Sixth ICICSE 2018*. Springer, 2019, pp. 295–305.

[6] D. Tourky, M. ElKawkagy, and A. Keshk, "Homomorphic encryption the "holy grail" of cryptography," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, pp. 196–201, accessed: 2024-03-06.

[7] B. Pulido-Gaytan, A. Tchernykh, J. M. Cortés-Mendoza, M. Babenko, G. Radchenko, A. Avetisyan, and A. Y. Drozdov, "Privacy-preserving neural networks with homomorphic encryption: C hallenges and opportunities," *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1666–1691, 2021.

[8] N. B. Njungle, E. Jahns, Z. Wu, L. Mastromauro, M. Stojkov, and M. Kinsy, "Guardianml: Anatomy of privacy-preserving machine learning techniques and frameworks," *IEEE Access*, 2025.

[9] C. Banerjee, T. Mukherjee, and E. Pasiliao Jr, "An empirical study on generalizations of the relu activation function," in *Proceedings of the 2019 ACM Southeast Conference*, 2019, pp. 164–167.

[10] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[11] R. Podschwadt, D. Takabi, P. Hu, M. H. Rafiei, and Z. Cai, "A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption," *IEEE Access*, vol. 10, pp. 117 477–117 500, 2022.

[12] D. Catalano, R. Cramer, G. Di Crescenzo, I. Darmgård, D. Pointcheval, T. Takagi, R. Cramer, and I. Damgård, "Multiparty computation, an introduction," *Contemporary cryptology*, pp. 41–87, 2005.

[13] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1651–1669. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar

[14] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 619–631.

[15] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "{XONN}:{XNOR-based} oblivious deep neural network inference," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1501–1518.

[16] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International conference on machine learning*. PMLR, 2016, pp. 201–210.

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

[18] 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 Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1330–1343, 2020.

[19] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," Cryptology ePrint Archive, Paper 2018/1041, 2018. [Online]. Available: https://eprint.iacr.org/2018/1041

[20] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," Cryptology ePrint Archive, Paper 2017/1114, 2017. [Online]. Available: https://eprint.iacr.org/2017/1114

[21] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: Fast fully homomorphic encryption over the torus," Cryptology ePrint Archive, Paper 2018/421, 2018, https://eprint.iacr.org/2018/421. [Online]. Available: https://eprint.iacr.org/2018/421

[22] A. A. Badawi and Y. Polyakov, "Demystifying bootstrapping in fully homomorphic encryption," Cryptology ePrint Archive, Paper 2023/149, 2023. [Online]. Available: https://eprint.iacr.org/2023/149

[23] H. Narumanchi, D. Goyal, N. Emmadi, and P. Gauravaram, "Performance analysis of sorting of fhe data: integer-wise comparison vs bit-wise comparison," in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2017, pp. 902–908.

[24] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," Cryptology ePrint Archive, Paper 2016/421, 2016. [Online]. Available: https://eprint.iacr.org/2016/421

[25] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, *et al.*, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *iEEE Access*, vol. 10, pp. 30 039–30 054, 2022.

[26] D. Kim and C. Guyot, "Optimized privacy-preserving cnn inference with fully homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2175–2187, 2023.

[27] L. Rovida and A. Leporati, "Encrypted image classification with low memory footprint using fully homomorphic encryption," Cryptology ePrint Archive, Paper 2024/460, 2024. [Online]. Available: https://eprint.iacr.org/2024/460

[28] E. F. Brickell and Y. Yacobi, "On privacy homomorphisms (extended abstract)," in *Advances in Cryptology — EUROCRYPT' 87*, D. Chaum and W. L. Price, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 117–125.

[29] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009, crypto.stanford.edu/craig.

[30] N. Aggarwal, C. Gupta, and I. Sharma, "Fully homomorphic symmetric scheme without bootstrapping," pp. 14–17, 2014.

[31] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 144, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:1467571

[32] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Procedings of the thirty-seventh annual ACM symposium on Theory of Computing*, 2005.

[33] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," Cryptology ePrint Archive, Paper 2012/230, 2012. [Online]. Available: https://eprint.iacr.org/2012/230

[34] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE Security & Privacy*, vol. 17, no. 2, pp. 49–58, 2019.

[35] Z. Ghahramani, *Unsupervised Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 72–112.

[36] T. Jiang, J. L. Gradus, and A. J. Rosellini, "Supervised machine learning: A brief primer," *Behav Ther*, vol. 51, no. 5, pp. 675–687, May 2020.

[37] K. Sivamayil, E. Rajasekar, B. Aljafari, S. Nikolovski, S. Vairavasundaram, and I. Vairavasundaram, "A systematic study on reinforcement learning based applications," *Energies*, vol. 16, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1996-1073/16/3/1512

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://papers.nips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[39] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222008426

[40] M. Roodschild, J. Gotay Sardiñas, and A. Will, "A new approach for the vanishing gradient problem on sigmoid activation," *Progress in Artificial Intelligence*, vol. 9, no. 4, pp. 351–360, 2020.

[41] Y. Hu, A. Huber, J. Anumula, and S.-C. Liu, "Overcoming the vanishing gradient problem in plain recurrent networks," *arXiv preprint arXiv:1801.06105*, 2018.

[42] N. Gupta, P. Bedi, and V. Jindal, "Effect of activation functions on the performance of deep learning algorithms for network intrusion detection systems," in *Proceedings of ICETIT 2019: Emerging Trends in Information Technology*. Springer, 2020, pp. 949–960.

[43] J. He, L. Li, J. Xu, and C. Zheng, "Relu deep neural networks and linear finite elements," *arXiv preprint arXiv:1807.03973*, 2018.

[44] L. A. Gil-Alana and J. C. Cuestas, "A non-linear approach with long range dependence based on chebyshev polynomials," 2012.

[45] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," Cryptology ePrint Archive, Paper 2016/870, 2016. [Online]. Available: https://eprint.iacr.org/2016/870

[46] C. Boura, N. Gama, M. Georgieva, and D. Jetchev, "CHIMERA: Combining ring-LWE-based fully homomorphic encryption schemes,"

Cryptology ePrint Archive, Paper 2018/758, 2018. [Online]. Available: https://eprint.iacr.org/2018/758

[47] A. A. Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca, "Openfhe: Open-source fully homomorphic encryption library," Cryptology ePrint Archive, Paper 2022/915, 2022, https://eprint.iacr.org/2022/915. [Online]. Available: https://eprint.iacr.org/2022/915

[48] A. Al Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca, "Openfhe: Open-source fully homomorphic encryption library," in *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, ser. WAHC'22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 53–63. [Online]. Available: https://doi.org/10.1145/3560827.3563379

[49] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[51] D. M. Jennewein, J. Lee, C. Kurtz, W. Dizon, I. Shaeffer, A. Chapman, A. Chiquete, J. Burks, A. Carlson, N. Mason, A. Kobawala, T. Jagadeesan, P. B. Basani, T. Battelle, R. Belshe, D. McCaffrey, M. Brazil, C. Inumella, K. Kuznia, J. Buzinski, D. D. Shah, S. M. Dudley, G. Speyer, and J. Yalim, "The sol supercomputer at arizona state university," ser. PEARC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 296–301. [Online]. Available: https://doi.org/10.1145/3569951.3597573