Enhancing Cloud Security through Efficient Polynomial Approximations for Homomorphic Evaluation of Neural Network Activation Functions

Bernardo Pulido-Gaytan[1], Andrei Tchernykh[1,4,*]
[1]*CICESE Research Center,*
Ensenada, Baja California, Mexico
{lpulido, chernykh}@cicese.edu.mx

Mikhail Babenko[2,4]
[2]*North-Caucasus Federal University,*
Stavropol, Russia
mgbabenko@ncfu.ru

Jorge M. Cortés-Mendoza[3], Horacio González-Vélez[3]
[3]*National College of Ireland,*
Dublin, Ireland
{JorgeMario.CortesMendoza, horacio}@ncirl.ie

Arutyun Avetisyan[4]
[4]*Institute for System Programming, RAS,*
Moscow, Russia
arut@ispras.ru

***Abstract*—Current security cloud practices can successfully protect stored data and data in transit, but they do not keep the same protection during data processing. The data value extraction requires decryption, creating critical exposure points. As a result, privacy-preserving techniques are emerging as a crucial consideration in cloud computing. The homomorphic processing of machine learning models in the cloud represents a central challenge. The activation function is fundamental in constructing a privacy-preserving Neural Network (NN) with Homomorphic Encryption (HE). Standard activation functions require operations not supported by HE, so it is necessary to find cryptographically compatible replacement functions to operate over encrypted data. Multiple approaches address the limitation of function compatibility with polynomial approximation. These functions should exhibit a trade-off between complexity and accuracy, limiting the efficiency of conventional approximation techniques. The current literature on polynomial approximation of NN activation functions still lacks a thorough review. In this paper, we comprehensively review the standard activation functions of modern NN models and current polynomial approximation approaches. We highlight fundamental features to consider in the activation function and the approximation technique to operate over encrypted data.**

Keywords—cloud security, homomorphic encryption, neural networks, polynomial approximation, privacy-preserving.

I. Introduction

The data processing is commonly overlooked in the data life cycle of modern security cloud practices. Conventional cryptosystems successfully protect stored data and data in transit. However, they do not provide the same properties during data processing when data decryption creates critical exposure points. As a result, privacy-preserving techniques are emerging. The general idea consists of delegating data processing without giving transparent access to it.

Machine Learning as a Service (MLaaS) is a flexible and scalable solution to train and deploy predictive models in cloud environments [1]. However, a critical limitation of the extensive adoption of MLaaS is the low protection of sensitive information. In most cases, the models require full access to confidential data, which generates privacy concerns.

Incorporating Homomorphic Encryption (HE) into MLaaS is an elegant way to solve this apparent paradox. For instance, a Neural Network (NN) model with HE can provide privacy-preserving in predicting or classifying confidential information. On the one hand, NN models are extensively used due to their remarkable results in multiple domains. On the other hand, HE enables the processing of encrypted information by a non-trustworthy third party without disclosing confidential data. Hence, combining their potentialities allows a system to provide privacy, accuracy, scalability, and flexibility.

HE is a family of encryption schemes that perform computations on ciphertexts. The arithmetic operations applied to ciphertext yield the same result as applying the operations to the original unencrypted data. Nonetheless, current HE schemes support only additions and multiplications. It restricts its adoption in many fields and hinders the NN construction in its commonly known form.

The main challenge in developing NNs with HE (NN-HE) is the computational design of cryptographically computable neuron functions for homomorphic processing, mainly the activation function [2]. The activation function is an essential element in constructing an effective NN. It determines the accuracy and computational efficiency of the NN. Moreover, activations significantly affect the model converging speed.

The implementation of standard activation functions needs operations beyond additions and multiplications. For instance, computing the Rectified Linear Unit (ReLU) requires a comparison operator. Hence, HE constraints demand finding compatible replacement functions to operate over encrypted data. Multiple approaches address the limitation by polynomially approximating non-compatible functions with a cryptographically consistent polynomial form.

The design of an adequate polynomial approximation function depends on the balance between complexity and accuracy. A high degree polynomial provides a more accurate approximation but slows down the computations performed on encrypted data. In practice, an inadequate approximation function can result in poor performance and long processing time of the NN-HE.

Current literature shows that standard activation functions can be substituted by polynomials from different degrees [3], [4]. However, they are still solutions to specific problems, so no generic approach exists. The literature in polynomial approximation for privacy-preserving NN activation functions still lacks a thorough review.

In this paper, we comprehensively review the standard activation functions of modern NNs. To construct an efficient NN-HE that operates over encrypted data, we also present current polynomial approximation approaches. Furthermore, we highlight fundamental aspects to consider in selecting the activation function and polynomial approximation techniques.

The content of the paper is structured as follows. Section 2 addresses privacy-preserving NN-HE. Section 3 describes the main activation functions. Section 4 presents relevant works. Section 5 reviews the state-of-the-art polynomial approximation approaches. Section 6 discusses fundamental features to consider in constructing cryptographically computable activation functions. Finally, we conclude and discuss future works in Section 7.

II. Privacy-Preserving Neural Networks

A promising long-pursued application is a privacy-preserving NN-HE. Current literature has shown remarkable advances in the field [2], [5]. This section addresses the importance and shortcomings of the homomorphic evaluation of complex NNs. First, we provide basic notions in NNs. Later, we present relevant aspects of NN-HE implementation.

A. Neural Networks

In the last decades, NNs have been the primary research direction of ML models. They provide suitable solutions in a wide field of human knowledge. Its massive adoption and generalized use reveal several data privacy problems in the model construction. An underlying notion of NN is required to clarify the privacy concerns. A NN is a computing model system that mimics a biological brain; its main goal is to identify underlying relationships in information. The neurons are basic units that process information coming from the external world.

Each neuron consists of k inputs $x_1, \ldots, x_k$ and an output $f(y)$, where $y = \sum_{i=1}^{k} w_i x_i + \beta$. The value of y defines a weighted-sum of the inputs with respect to the synaptic weights $w_1, \ldots, w_k$ and a bias β. The non-linear activation function f generates the output of the neuron. The definitions of y and f are fundamental problems in NN research.

The interaction between neurons is essential in NN performance. The NN architecture groups neurons in layers and establishes the connection between them. The position of layers in the sequence of the NN establishes a specific role in data processing. In this way, a standard classification of layers in a NN comprises three types. The first layer in the NN is the input layer. It receives information from outside the network. Internal layers, also called hidden, are not directly accessible from the exterior. The last segment in the NN is the output layer. It transfers information outside of the network. Neuron connections can involve neurons of the same layer (self-recurrent), a successive layer (feedforward), or a previous layer (recurrent). This paper focuses on feedforward NNs, where each neuron is evaluated only once; they are more widely used and more straightforward to evaluate homomorphically.

The activation function f determines the neuron reaction to the inputs and the information forward to the following layers. The most successful and widely-used activation functions are Sigmoid, Tangent Hyperbolic (Tanh), and ReLU. All of them have some advantages and disadvantages in different domains. The activation function is essential in the construction of an efficient NN. It contributes to extracting the underlying relationships in the data by the learning process.

The training phase consists of developing a mapping from the input to the output space by adjusting the weights of each neuron. An example set allows the NN to learn and generalize information. The intervention of an external entity that provides correct output labels to guide the learning process is defined as Supervised Learning (SL). Meanwhile, its absence is denoted as Unsupervised Learning (UL). In SL, weights are modified when the expected output $\hat{\vartheta}$ and the actual NN output ϑ are different; the loss function $L(\hat{\vartheta}, \vartheta)$ measures the error between both outputs. The aim is to find a set of weights that minimize L.

The learning acquired in the training phase enables the NN to provide correct answers in unknown patterns. In the testing phase, the NN is evaluated under an example set distinct to the training phase, avoiding overfitting. The general process is an analogy of the brain evolution during a human experience. A significant disadvantage of NN models is related to the training process. It can consume many computational resources and time with relatively big datasets.

The use of high-performance computing resources can reduce training and testing time. However, this kind of infrastructure is not easily available due to the high deployment and maintenance costs. In recent years, cloud services have emerged as an alternative to solve the need for high-capacity infrastructure [6]; the Cloud Service Providers (CSPs) provide resources and tools to deploy ML models remotely.

The remote third-party infrastructure of the cloud reduces resource and implementation problems, but it introduces several privacy concerns for sensitive information [7]. The data can be protected by encryption pre- and post-processing, but during the processing, the NN needs access to raw data, which creates potential privacy risks. The following section describes a solution by allowing encrypted data to be processed by a remote server.

B. Homomorphic Evaluation of NN Models

Legal and ethical requirements may prevent the adoption of cloud-based solutions for many ML applications [8], [9]; accessing the raw data for training and testing a model can have several privacy consequences. For example, the processing of DNA sequencing remains a complex task, but new technologies and methods make it faster and less expensive [10]. In this situation, the potential risks and benefits should be evaluated. Sharing and processing the genetic sequence with ML models could offer significant value, such as pathology diagnosis. However, the DNA sequence contains partial information about the DNA of the relatives; it falls into severe problems if such information reaches unreliable third parties. This scenario shows a common concern of data privacy in different domains: medical

and financial, among others. These problems detonated the interest of evaluating complex NNs over encrypted data.

One strong direction focuses on the benefits of HE schemes for information processing in a NN [3]. An NN-HE is a natural extension of NN models; the methodology consists of applying HE to the network inputs and propagating the data across the network homomorphically. However, there are some problems to solve in designing an efficient NN-HE.

The main challenge of NN-HE is the homomorphic design of the neurons. The restricted number of operations in HE schemes limits the implementation of inner functions of neurons to polynomials. Additionally, the multiplication operation is slower and adds a large amount of noise. Likewise, bootstrapping is a time-consuming operation that should be performed to reduce the noise in the ciphertext after a certain number of operations [11].

Each neuron performs a *weighted-sum* y and a non-linear *activation function* f. The computing of y consists of a set of additions and multiplications between constant weights and ciphertexts; it can be improved since one of the operands is plaintext, and the size of the resultant ciphertext remains the same as the input ciphertext [12]. The optimized solution to compute an operation between plaintext and ciphertext consists of encrypting the constant without noise and thus performing the standard homomorphic operation.

The processing of f in HE is more complicated because most functions are not polynomials. Several approaches replaced standard functions with the non-linear low-degree square function [3], [13]. Nonetheless, the unbounded derivate of the square function induces a strange behavior during the training phase; it has been demonstrated in NNs with more than two non-linear layers [14].

Multiple approaches address the limitation of the activation function through polynomial approximation [14], [15], [16], [17]. Since all NN inner functions are continuous, the NN itself can be viewed as continuous. Also, if the domain (input space) is a compact set, then the Stone-Weierstrass theorem [18] establishes that it can be approximated uniformly by polynomials. The challenge consists of approximating the NN with polynomials of the lowest possible degree.

Let N be a network with l layers. If a polynomial of degree n approximates the activation function in each layer, then the approximation of N will be a polynomial of degree n^l. Thus, to achieve a low-degree polynomial, n and l must be small. Even so, optimizing l implies reducing the layers in N. The following sections address the exercise of minimizing n.

III. Activation Functions

The activation function is fundamental in the convergence speed and efficiency of the NN. The derivative of the activation function, also known as the gradient, is essential for the training phase. Both determine and normalize the output of the NN model, its accuracy, and computational efficiency. There are three main activation function types: binary step, linear, and non-linear functions.

A *binary step* activation function defines a threshold t to activate the neuron. The neuron propagates a value when the input value is above or below t, with $f(y) = 0$ if $y < t$ and $f(x) = 1$ otherwise. However, a step function does not support multi-value outputs, i.e., it cannot classify inputs into one of several classes.

A *linear* activation function returns an output directly proportional to the input argument. It solves the disadvantage of multi-value outputs but introduces two significant problems. First, the NN cannot be trained using backpropagation algorithm. The constant value of the derivative does not exhibit any relation to the input. Hence, it is not possible to discern which weights contribute to better accuracy. Second, a linear activation function turns the NN into a one-layer network, i.e., a linear regression model. All layers collapse into one because a combination of linear functions generates a linear function, where the last layer will be a linear function of the first layers.

Modern NN models use *non-linear* activation functions. They allow the creation of complex mappings between the inputs and outputs of the network, which are essential for learning and modeling complex data with high dimensionality. Non-linear functions support backpropagation and "stacking" of layers. Multiple hidden layers are needed to learn complex datasets with high levels of accuracy. The non-linearity notion in NNs is introduced by a non-linear function that serves an integral role in the training and performance evaluation. Conveniently, any mention of an activation function in this manuscript will refer to a non-linear function.

Several activation functions have been proposed in the literature, but only a few are widely used [19], [20]. The following sections focus on standard non-linear functions, emphasizing the advantages and disadvantages of Sigmoid, Tanh, ReLU, Leaky ReLU (LReLU), and Swish, see Fig. 1.

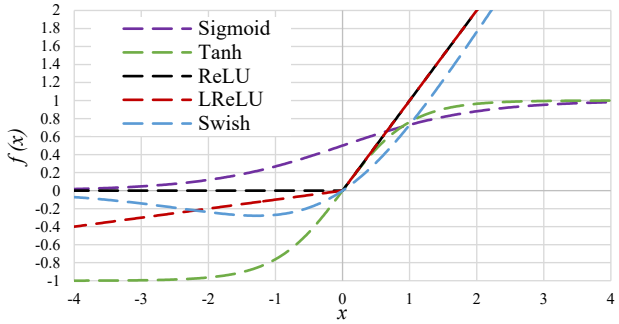

Fig. 1 Activation functions

A. Sigmoid

The Sigmoid function is a fundamental part of a Gradient Descent (GD) algorithm that cannot be implemented on HE cryptosystems, see (1).

$$f(x) = \frac{1}{1+e^{-x}} \tag{1}$$

Sigmoid is a monotonic, bounded, and smooth activation function. *Smoothness* plays a beneficial role in optimization and

generalization. A smooth function is more traversable, reducing sensitivity to initialization and learning rates.

The bounding at the top and bottom of the Sigmoid allows us to normalize the neuron output values in the range $[0, 1]$. However, the *unboundedness* feature is desirable because it prevents saturation as training becomes slow near-zero gradient values [21]. Also, the NN must be carefully initialized to stay in the linear regime of the function. Sigmoid tends to bring the predictions to the curve edge, close to 1 or 0. However, prediction for high or low values almost does not change; it causes a problem in the training phase named the Vanishing Gradient Problem (VGP).

B. Tanh

Tanh is another classic activation function defined as

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{-x} + e^{x}} \quad (2)$$

Tanh inherits the limitations and advantages of Sigmoid. Tanh is distinguished by being *zero-centered*, making it easier to model inputs with strongly negative and positive values, and addressing the VGP.

C. ReLU

Currently, the most successful and widely-used activation function is the ReLU [22], [23], defined as

$$f(x) = \max(x, 0) \quad (3)$$

ReLU was a breakthrough that enabled the fully supervised training of state-of-the-art Deep Neural Networks (DNN) [24]. It is the default activation function in almost all applications due to its simple implementation and effectiveness concerning other activations. ReLU is a simple piecewise function: the output is zero when the input x is negative, and the output is x when $x > 0$. Thus, ReLU has a gradient of one for positive and zero for negative inputs. The gradient preserving feature allows NNs with ReLU to be more easily optimized than NNs with Sigmoid or Tanh.

Sigmoid and Tanh were improved by the unbounded ReLU. This feature helps in strong regularization effects because the function avoids saturation whenever $x > 0$. Functions with an approach to zero in the limit induce even larger regularization effects since large negative inputs are forgotten.

Nonetheless, the NN cannot perform backpropagation and learn when the input values are zero or negative; the gradient of ReLU becomes zero. This undesirable situation leads to slow training or halts for units that receive mostly negative inputs. Weights feed into units with a zero gradient stop training, so portions of the network "die" during training.

D. Leaky and Parametric ReLU

LReLU and Parametric ReLU (PReLU) variations address the ReLU limitations. LReLU allows a small, non-zero, constant gradient α instead of zero when $x < 0$,

$$f(x) = \max(x, \alpha x) \quad (4)$$

LReLU presents a small positive slope in the negative area, enabling backpropagation, even for negative values. This change improves accuracy over standard function by preventing units from having a zero gradient.

PReLU is a dynamic version of LReLU; it allows the negative slope to be learned for each unit during training. This function provides the negative slope part of the function as an argument. Hence, it is possible to perform backpropagation and learn the most appropriate value. However, such dynamism makes PReLU susceptible to overfitting problems.

E. Swish

Swish [23] is a smooth, unbounded, self-gated, non-monotonic activation function defined as

$$f(x) = x \cdot \sigma(x), \quad (5)$$

where $\sigma(x)$ denotes the Sigmoid function, see (1). *Self-gating* is a technique inspired by using the Sigmoid function in Long Short-Term Memory (LSTM). An advantage of self-gating is that it only requires a single input, whereas normal gates require multiple scalar inputs. Thus, Swish can replace ReLU as it also takes only a single scalar input (pointwise functions) without changing the hidden size or number of parameters. Swish is unbounded above and bounded below. Unlike ReLU, Swish is smooth and non-monotonic. Swish's non-monotonic property distinguishes itself from the standard activation functions. Swish can perform better than ReLU with a similar level of computational efficiency [23].

The feature of *approaching zero* in the limit is satisfied by Tanh, ReLU, and Swish. Nonetheless, Swish differs because it produces negative outputs for small negative inputs due to its non-monotonicity. Non-monotonicity increases expressivity and improves gradient flow. This property also provides robustness to different initializations and learning rates.

Avenash and Viswanath [25] present Hard-Swish, which improves Swish performance because it does not involve any exponential operation. Hard-Swish is defined as $f(x) = \max(0, \min(1, (\delta x * 0.2 + 0.5)))$, where δ is either a trainable or constant parameter. However, combining max and min comparisons complicates its polynomial approximation and, thus, its homomorphic evaluation.

Table I summarizes the most important aspects of activation functions.

TABLE I. SUMMARY OF ACTIVATION FUNCTIONS

Approximation Function / Feature	*Sigmoid*	*Tanh*	*ReLU*	*(L-P) ReLU*	*Swish*
Differentiable	•	•	•	•	•
Smoothness	•	•			•
Self-gating	•	•			•
Approaching zero		•	•	•	•
Non-monotonic					•
Non-monotonic derivative	•	•			•
Unbounded			•	•	•
$f(x)$	$[0,1]$	$[-1,1]$	$[0,\infty)$	$(-\infty,\infty)$	$[\approx -0.278,\infty)$

IV. Related Work

Several approaches to approximate activation functions in multiple domains have been studied. This section presents relevant works in the polynomial approximation of common activation functions.

The problem of Sigmoid approximation has been widely studied in homomorphic Logistic Regression (LR) models with GD [26]. The LR model is the simplest form of the NN, comprising a single layer without hidden layers and logistic activation functions. While the NN complexity is directly proportional to the number of layers and activation functions, the fundamental methods of activation function approximation by polynomials are the same in LR and NN models.

Aono et al. [27] propose a Homomorphism-Aware LR (HA-LR) model to protect data under encryption. The authors study the quality of the model with several Somewhat Homomorphic Encryption (SHE) schemes in the training and testing phases. HA-LR provides better accuracy than a non-homomorphic LR for the Pima dataset (between 0.27% and 0.57%) and a worse accuracy related to the SPECTF dataset, within 2.17% to 3.7%.

Bonte et al. [28] develop a Privacy-Preserving LR (PP-LR) model with SHE. A central server concentrates the data of several users and trains a binary classification model without learning anything about the underlying information. The accuracy of PP-LR is only slightly worse than the non-homomorphic LR version, considering real-life problems and several accuracy metrics.

Kim et al. [29] present a method to train an LR model with a SHE scheme (SHE-LR). The authors improve the convergence of GD by implementing the Nesterov algorithm and reducing encrypted data storage with a novel packing method. SHE-LR significantly reduces the training time to only 3%, on average, with respect to other approaches and provides similar prediction values considering two metrics.

Chabanne et al. [14] present a DNN evaluated homomorphically. The network replaces ReLU with a low-degree polynomial. A batch normalization layer is added before each activation layer to have a restricted stable distribution at the ReLU inputs, limiting the need for an accurate approximation to a bounded interval. The authors used the polynomial regression function *polyfit* from the Python package NumPy [30].

Hesamifard et al. [15] develop a privacy-preserving DNN by replacing standard non-linear activation functions with HE-friendly low-degree polynomial approximations within a specific error range. Instead of approximating the ReLU itself, CryptoDL simulates the structure of the ReLU derivative. For Sigmoid and Tanh, the authors use the Chebyshev series over a symmetric interval.

Chou et al. [16] develop NN-HE using sparse representations throughout the NN to accelerate homomorphic evaluations. They derive an approximation with maximally sparse encodings through a polytope-based method [31]. Experiments show that NN-HE maintains competitive accuracy and achieves a significant speedup over previous methods.

Liao et al. [32] implement NN-HE to make predictions over encrypted sensor data. The authors approximated Tanh, ReLU, and Swish activation functions following the CryptoDL approach. The authors show that NN-HE accuracy is worse than the non-homomorphic version in one of six values and better only in two cases.

Timmons and Rice [33] use approximation techniques to develop better-performing drop-in replacements for existing NN functions. The authors show that the approximations can improve the training and testing time without affecting the NN accuracy. Despite not implementing a HE scheme, the operations over polynomials make it feasible.

Khan and Michalas [34] propose a hybrid ML model named Learning in the Dark (LitD). The training phase of LitD occurs in plaintext data, and the classification process is performed directly on ciphertexts. LitD approximates the ReLU and Sigmoid activations using low-degree Chebyshev polynomials to classify images with high accuracy and reduced time.

Table II summarizes the most important characteristics of the cryptographically non-computable activation functions approximated by polynomials.

TABLE II. Summary of Activation Function Approximations

Function	*n*	Model		Approximation Method	Ref.
		LR	NN		
Sigmoid	2, 3		•	Chebyshev polynomials	[15]
	2	•		Taylor series, area	[27]
	1	•		Taylor series	[28]
	3, 5, 7	•		Taylor series	[29]
	9		•	Taylor series, Padé	[33]
	3,5,7,9		•	Chebyshev polynomials	[34]
Tanh	2, 3		•	Chebyshev polynomials	[15]
	9		•	Taylor series, Padé	[33]
	3, 4		•	Chebyshev polynomials	[32]
ReLU	2,3,4,5,6		•	Least squares polynomial fit (soft.)	[14]
	2, 3		•	Derivative of ReLU	[15]
	1		•	Taylor series, Padé	[33]
	3, 4		•	Chebyshev polynomials	[32]
	2		•	Polytope-based method	[16], [31]
	3,5,7,9		•	Chebyshev polynomials	[34]
Swish	3,4		•	Chebyshev polynomials	[32]
	2		•	Polytope-based method	[16], [31]

We provide an example of polynomial approximation to the Sigmoid function. The approximate polynomials to $f(x) = 1/(1+e^{-x})$ of degrees one, three, five, and seven [28], [29] are defined as follows:

$$f_1(x) = 0.5 + 0.25x \tag{6}$$

$$f_3(x) = 0.5 + (1.20096)\frac{x}{8} - (0.81562)\left(\frac{x}{8}\right)^3 \tag{7}$$

$$f_5(x) = 0.5 + (1.53048)\frac{x}{8} - (2.3533056)\left(\frac{x}{8}\right)^3 + (1.3511295)\left(\frac{x}{8}\right)^5 \tag{8}$$

$$f_7(x) = 0.5 + (1.73496)\frac{x}{8} - (4.19407)\left(\frac{x}{8}\right)^3 + (5.43402)\left(\frac{x}{8}\right)^5 - (2.50739)\left(\frac{x}{8}\right)^7 \tag{9}$$

Fig. 2 shows the Sigmoid function and the approximations in the interval [-4, 4]. We can see that the approximation degree is proportional to its accuracy; a more precise approximation implies a higher degree. The number of operations depends on

the polynomial degree, so the degree is related to the execution time to evaluate a model.

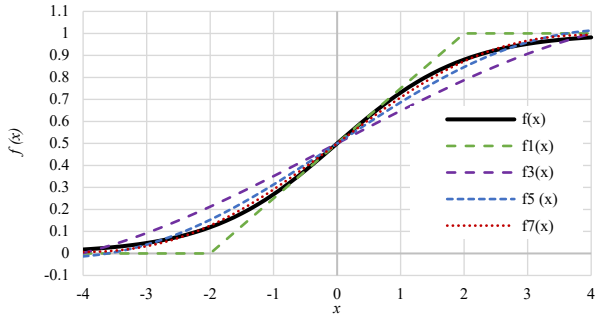

Fig 2. Example of polynomial approximation for the Sigmoid function.

Table III presents the maximum approximation error of the functions in the interval [-4, 4]. Low-degree polynomials have a bigger error concerning the original function, i.e., the area between the original and the approximated function is higher.

TABLE III. MAXIMUM APPROXIMATION ERRORS.

Approximation	Interval $[-4, 4]$
f_1	1.50033
f_3	0.11432
f_5	0.04708
f_7	0.03214

The following section presents several methods to find a suitable approximation.

V. POLYNOMIAL APPROXIMATION

Polynomials are the most well-behaved and straightforward to compute among continuous functions. A basic property of a polynomial $p(x) = \sum_{i=0}^{n} c_i x^i$ is that its value for a given x can be calculated in a finite number of steps. A central problem of mathematical analysis is the approximation of more general functions by polynomials and how small the discrepancy can be made. This section introduces a theoretical foundation of polynomial approximation and discusses several methods proposed in the literature.

A. Theoretical Base

Let us denote the set of all continuous real-valued functions $f: X \to \mathbb{R}$ on a non-empty compact space X by $C(X, \mathbb{R})$. For any $f, g \in C(X, \mathbb{R})$, $x \in X$, and $c \in \mathbb{R}$, let $f + g$ be defined by $(f + g)(x) = f(x) + g(x)$ and $c \times f$ by $(c \times f)(x) = c \times f(x)$. Since both $f + g$ and $c \times f$ are continuous if f, g are continuous, $C(X, \mathbb{R})$ forms a normed space over $\mathbb{R}$.

We say that f is $C([a, b], \mathbb{R})$, meaning that f is continuous for $a \leq x \leq b$. Conveniently, we define X as a closed, bounded interval $[a, b]$ in $\mathbb{R}$, i.e., X is $C([a, b], \mathbb{R})$. Now, let us denote a subset $A \in C(X, \mathbb{R})$ as the set of polynomials in a single variable with real coefficients. Since linear combination and product of polynomials are also polynomials, we assume that A is closed under addition, scalar multiplication, and product, and a non-zero constant function belongs to A.

Hence, an element $f \in C(X, \mathbb{R})$ can be approximated by elements of A, if for every $\epsilon > 0$, there exists $p \in A$ such that $|f(x) - p(x)| < \epsilon$ for every $x \in X$. The following theorem establishes that every continuous real-valued function on a non-empty compact interval $f \in C(X, \mathbb{R})$ can be uniformly approximated by elements of A.

Theorem 1 (Stone-Weierstrass). *Every element of $C(X, \mathbb{R})$ can be approximated by elements of A if and only if for every $x \neq y \in X$, there exists $p \in A$ such that $p(x) \neq p(y)$.*

In other words, the polynomials are uniformly dense in $C([a, b], \mathbb{R})$ with the supremum norm. The supremum norm of a function $f \in C(X, \mathbb{R})$ is given by $\|f\|_u = \sup|f(x)|$. Since X is compact and f is continuous, the supremum is always guaranteed to exist. Thus, for every $f \in C(X, \mathbb{R})$ and a given ϵ, we can find $p(x)$ such that

$$\sup|f(x) - p(x)| < \epsilon \tag{10}$$

The goodness of the fit of a particular polynomial $p(x)$ to the function $f(x)$ be measured by $\sup|f(x) - p(x)|$. For more details on the subject, refer to [35].

The polynomial approximation is a standard exercise in approximation theory. Hence, multiple approaches are found to approximate a given function by polynomials, from simple to theoretically well-founded techniques. The main shortcoming of simple analytic methods is poor performance. The following sections present two widely used approximation techniques.

B. Taylor Series

Taylor series is a popular approach for accurately approximating arbitrary functions on a specific interval. A method requirement is the analytic property of the function (highly differentiable), but the subclass of functions with this property is relatively small. A common query is the sufficiency of this condition to approximate a function with an arbitrary polynomial degree on a closed interval.

The method suffers from two limitations with high repercussions on the HE domain. The first one is the degree of polynomial approximation. Due to their high degree, these polynomials cannot be efficiently computed in HE schemes. Second, the approximation interval. The Taylor method approximates a function in the neighborhood of a point; the approximation error is higher in points not included in the input interval. The Taylor series expansion can have problems with image datasets. It cannot cover the MNIST dataset elements that contain integer values for pixels in the interval [0, 255].

C. Chebyshev Polynomials

Chebyshev polynomials are a family of orthogonal polynomials with a recursive definition. They mainly concentrate on the interval endpoints, which causes interpolation at mostly initial and endpoints, and less oscillation in the approximation. A Chebyshev approximation of the activation function in a large interval can avoid extra layers. For more details on orthogonal polynomials, refer to [36].

Moreover, the Chebyshev approximation allows that if the function to be approximate is piecewise linear, the Chebyshev series still converges, i.e., is still applicable. That is the case of ReLU, which is non-smooth on zero.

Let p_n the set of polynomials of degree less than or equal to n, where for any $p \in p_n$, $d(p, f) = \sup|f(x) - p(x)|$ for $a \leq$

$x \leq b$, with $d \geq 0$. The method establishes that exists a $p \in p_n$ for which the $\inf d(p, f)$ is attained, i.e., given a $f \in C([a, b], \mathbb{R})$, there is a unique polynomial of degree n of best approximation. Nonetheless, calculating the best p for a function f is only practicable in exceptional cases.

Replacing a function with a low-degree polynomial is an intricate task. The nature of LR or NN models further complicates it. An accurate approximation for homomorphic evaluation of NN activations is directly related to factors such as the degree, error, interval, and coefficient precision. The following section delves into these important features.

VI. Polynomial Approximation in Privacy-Preserving Neural Networks

Using a cryptographically computable activation function approximation reduces the inference accuracy of the model. This section highlights fundamental features in the polynomial approximation of privacy-preserving activation functions to operate over encrypted data.

A. Degree

An activation function can be approximated with polynomials of different degrees. The design of an adequate approximation depends on the balance between complexity and accuracy [26], [29]. A high-degree polynomial offers a more accurate approximation and, thus, better NN accuracy. However, when operations are performed over encrypted data, a higher degree leads to high overhead. Low-degree polynomials provide a shorter evaluation time at the cost of increasing the relative error to original function, i.e., the approximation error.

Additionally, a high degree leads to large encryption parameters due to the large multiplicative depth. HE construction requires a polynomial that finds a trade-off between the degree of the approximation and the model performance.

B. Error

Polynomial activation layers in the NN reduce inference accuracy since the approximation errors lead to a low distortion of the output distribution of the next normalization layer. Errors in data processing can compromise the correctness of the results. The implementation of NN-HE distinguishes two classes of errors: running and algorithmic.

Running errors. The calculation errors can arise due to the polynomial approximation of activation functions. This error denotes the area between the original and the approximated function. For instance, let us consider $\mathrm{ReLU}(x) = \max(x, 0) = (\mathrm{sgn}(x) + 1) \cdot x$. The function $\max(a, b)$ returns the maximum value of a and b, where $\mathrm{sign}(x)$ is a sign function. When we calculate $\mathrm{ReLU}(x)$ for $x < 0$ over HE-encrypted data using the $\mathrm{sign}(x)$ algorithm presented by Cheon et al. [37], the function will be greater than zero. Thus, the implementation of NN-HE should be considered rounding errors that may occur.

Algorithmic errors. The representation of real numbers by integer polynomials can lead to an incorrect result. An error can occur in numbers near zero, even without adding noise. Converting a value vector to polynomial $p(X)$ can lead to a calculation error. For example, let us consider the Cheon-Kim-Kim-Song (CKKS) [38] scheme: $M = 8$ (i.e., the M-th cyclotomic polynomial $\Phi_8(X) = X^4 + 1$ of degree $n = \Phi_8(X)$) and a scaling factor $\Delta = 64$. Let $T = \{\xi_8, \xi_8^3\}$ for the root of unity $\xi_8 = \exp(2\pi i/8)$. For a given vector $z = (0.1, -0.01)$, the corresponding real polynomial $-0.039X^3 + 0.039X + 0.045$ (interpolation polynomial in the Lagrange form for a given set of points $(\xi, 0.1)$, $(\xi^3, -0.01)$, $(\overline{\xi}, \overline{0.1})$, $(\overline{\xi^3}, \overline{-0.01})$, where $\overline{a + bi} = a - bi$, $a, b \in R$ and $i^2 = -1$). Then, the output of the encoding algorithm is $p(X) = -2X^3 + 2X + 3$. Note that $64^{-1} \cdot (p(\xi_8), p(\xi_8^3)) \approx (0.09107, 0.00268)$ is approximated to the input vector z with high precision.

The 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. Consequently, an incorrect result using a NN-HE is highly probable when the input data are normalized. Moreover, this error leads to incorrect results when using unstable algorithms.

C. Interval

Another critical factor when approximating a function by polynomials is the interval. An approximation that does not consider the full range of values the activation function takes will lead to poor accuracy. The interval that offers the best accuracy is directly proportional to the values range in the dataset and the activation function normalization. Hence, this leads to another trade-off. A long interval requires a high-degree polynomial to be properly approximated. Therefore, the polynomial approximation is more accurate at small ranges.

Likewise, a high-degree polynomial and a long interval require highly precise coefficients. When the degree increases, the coefficients become very small and must be truncated to specific decimal places. Therefore, higher precision and accuracy require pruning the fewest decimal places.

VII. Conclusions

Homomorphic privacy preservation is an intensively developing research area with innumerable potential and commensurable benefits. Homomorphic evaluation of neural networks solves security and privacy concerns but implies redesigning their inner functions. Several limitations arise in reproducing the behavior of neurons using polynomial approximations in the homomorphic space.

Known cryptographically computable activation functions reduce the accuracy with respect to its original version. Unlike the traditional model, a suitable polynomial activation function should consider additional elements to improve accuracy. The polynomial degree, approximation error, interval, and precision of the coefficients are fundamental aspects of practical homomorphic evaluation. In practice, an adequately approximated function should increase the accuracy, reduce processing time, and improve the converging speed of the network. There is no generic approach. The literature in polynomial approximation for privacy-preserving NN activation functions still lacks a thorough review.

This paper highlights fundamental features, challenges, and solutions associated with the polynomial approximation of activation functions to operate over encrypted data. We review the standard non-linear activations of modern NN, emphasizing

the advantages and disadvantages of Sigmoid, Tanh, ReLU, Leaky ReLU, and Swish. We discuss their polynomial approximations, introduce a theoretical foundation, and present several methods proposed in the literature considering Taylor series and Chebyshev polynomials, which provide a solid basis for the state-of-the-art solutions in the area.

We discuss how approximation functions can balance complexity, accuracy, and multiplicative depth and analyze the importance of the interval when approximating a function by polynomials. We also consider errors in data processing, distinguishing two classes: running and algorithmic errors. The main lines of future work are related to the optimization of the encrypted number comparison to support a broader spectrum of activations in homomorphic NN models.

Acknowledgment

This research was funded by the Irish Research Council under grant GOIPD/2023/1341, "CA19135 - Connecting Education and Research Communities for an Innovative Resource Aware Society (CERCIRAS)," under STSM no. E-COST-GRANT-CA19135-f4613138; "DIGITAL4Security: European Masters Programme in Cybersecurity Management & Data Sovereignty" project (EU Digital Europe Programme grant number 101123430), and Ministry of Science and Higher Education of the Russian Federation (Project 075-15-2022-294).

References

[1] T. Hunt, C. Song, R. Shokri, V. Shmatikov, E. Witchel, "Chiron: Privacy-preserving machine learning as a service," arXiv:1803.05961, 2018.

[2] B. Pulido-Gaytan et al., "Privacy-preserving neural networks with Homomorphic encryption: Challenges and opportunities," Peer Peer Netw Appl, vol. 14, pp. 1666–1691, 2021, doi: 10.1007/s12083-021-01076-8.

[3] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in 33rd Int. Conf. on Machine Learning, 2016, pp. 201-210.

[4] T. Shortell and A. Shokoufandeh, "Secure signal processing using fully homomorphic encryption," in Advanced Concepts for Intelligent Vision Systems Conference, 2015, pp. 93–104.

[5] L. B. Pulido-Gaytan, A. Tchernykh, J. M. Cortés-Mendoza, M. Babenko, and G. Radchenko, "A survey on privacy-preserving machine learning with fully homomorphic encryption," in Latin America High Performance Computing Conference (CARLA), 2020, pp 93–104.

[6] M. Babenko et al., "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), 2021, pp. 548–553.

[7] A. Tchernykh et al., "Data reliability and redundancy optimization of a secure multi-cloud storage under uncertainty of errors and falsifications," in 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2019, pp. 565–572.

[8] A. Tchernykh, U. Schwiegelsohn, E. G. Talbi, and M. Babenko, "Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability," J Comput Sci, vol. 36, p. 100581, 2019, doi: 10.1016/j.jocs.2016.11.011.

[9] V. Miranda-López, A. Tchernykh, M. Babenko, A. Avetisyan, V. Toporkov, and A. Y. Drozdov, "2Lbp-RRNS: Two-levels RRNS with backpropagation for increased reliability and privacy-preserving of secure multi-clouds data storage," IEEE Access, vol. 8, pp. 199424-199439, 2020, doi: 10.1109/ACCESS.2020.3032655.

[10] J. M. Kidd et al., "Mapping and sequencing of structural variation from eight human genomes," Nature, vol. 453, no. 7191, pp. 56–64, 2008.

[11] M. Babenko et al., "Positional characteristics for efficient number comparison over the homomorphic encryption," Program. Comput. Softw., vol. 45, pp. 532–543, 2019, doi: 10.1134/S0361768819080115.

[12] 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.

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

[14] 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.

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

[16] 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.

[17] M. Babenko et al., "Towards the sign function best approximation for secure outsourced computations and control," Mathematics, vol. 10, no. 12, p. 2006, 2022, doi: 10.3390/math10122006.

[18] M. H. Stone, "The generalized Weierstrass approximation theorem," Mathematics Magazine, vol. 21, no. 4, pp. 167–184, 1948.

[19] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions," Neural Networks, vol. 138, pp. 14–32, 2021, doi: 10.1016/j.neunet.2021.01.026.

[20] 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.

[21] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[22] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," Nature, vol. 405, p. 947, 2000, doi: 10.1038/35016072.

[23] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," arXiv:1710.05941, 2017.

[24] A. Krizhevsky and G. Hinton, Learning multiple layers of features from tiny images. Master Thesis: University of Toronto, 2009.

[25] R. Avenash and P. Viswanath, "Semantic segmentation of satellite images using a modified CNN with Hard-Swish activation function," in VISIGRAPP, 2019, pp. 413–420.

[26] J. M. Cortés-Mendoza et al., "Privacy-preserving logistic regression as a cloud service based on residue number system," in Proceedings of the Russian Supercomputing Days Conference, 2020, p. 161.

[27] Y. Aono, T. Hayashi, and L. T. Phong, "Scalable and secure logistic regression via homomorphic encryption," in 6th ACM Conference on Data and Application Security and Privacy, 2016, pp. 142–144.

[28] C. Bonte and F. Vercauteren, "Privacy-preserving logistic regression training," BMC Med. Genomics, vol. 11, no. 86, 2018, doi: 10.1186/s12920-018-0398-y.

[29] A. Kim, Y. Song, M. Kim, K. Lee, J. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," BMC Med. Genomics., vol. 11, pp. 23–31, 2018, doi: 10.1186/s12920-018-0401-7.

[30] S. V. D. Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," Comput Sci Eng, vol. 13, no. 2, pp. 22–30, 2011, doi: 10.1109/MCSE.2011.37.

[31] N. Brisebarre, J.-M. Muller, and A. Tisserand, "Computing machine-efficient polynomial approximation," Trans. Math. Softw, 2006.

[32] 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.

[33] N. G. Timmons and A. Rice, "Approximating activation functions," arXiv:2001.06370, 2020.

[34] T. Khan, A. Michalas, "Learning in the dark: Privacy-preserving machine learning using function approximation," arXiv:2309.08190, 2023.

[35] V. K. Dzyadyk and I. A. Shevchuk, Theory of uniform approximation of functions by polynomials, De Gruyter, 2008, doi: 10.1515/9783110208245.

[36] W. Han and K. Atkinson, Theoretical numerical analysis: A functional analysis framework. 2009.

[37] 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.

[38] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for the arithmetic of approximate numbers," in Int. Conf. on the Theory and Application of Cryptology and Information Security, 2017, pp. 409–437.