Cryptographically Computable Self-Learning Activation Functions for Privacy-Preserving Logistic Regression with Homomorphic Encryption

Bernardo Pulido-Gaytan[1]
[1]*National College of Ireland*
Dublin, Ireland
luisbernardo.pulidogaytan@ncirl.ie

Horacio González-Vélez[1]
[1]*National College of Ireland*
Dublin, Ireland
horacio@ncirl.ie

Andrei Tchernykh[2]
[2]*CICESE Research Center*
Ensenada, Mexico
chernyh@cicese.mx

***Abstract*—Efficient processing of sensitive data in cloud environments requires not only accurate analysis but also rigorous privacy guarantees. The new generation of cloud-based Machine Learning (ML) services therefore demands privacy-preserving techniques that allow robust data analysis and individual privacy to effectively coexist. Homomorphic Encryption (HE) addresses privacy concerns by enabling computations over encrypted data, providing security and confidentiality to users even when execution takes place on untrusted shared infrastructures. Lattice-based HE schemes rely on the hardness of the Ring Learning with Errors problem and natively support only addition and multiplication operations, which poses significant challenges for efficiently implementing non-linear models. In this paper, we design and experimentally evaluate a specialized approach based on cryptographically computable self-learning polynomials to build accurate, low-complexity, and privacy-preserving Logistic Regression (LR) models for confidential data processing. We propose a method to increase the accuracy and performance of privacy-preserving LR with HE (LR-HE) using Self-Learning Activation Functions (SLAFs). SLAFs are polynomials with trainable coefficients that are updated during training, together with synaptic weights, to learn task-specific and LR-specific features. Our results demonstrate the feasibility of applying trained, problem-specific polynomial activations to generate HE-compliant logistic functions, enabling efficient, accurate, and privacy-preserving LR solutions in privacy-sensitive cloud settings.**

***Keywords*—Homomorphic encryption, logistic regression, polynomial approximation, privacy-preserving, self-learning activation functions.**

I. Introduction

The widespread adoption of cloud computing has transformed the deployment of large-scale Machine Learning (ML) models by offering scalable storage, on-demand computation, and cost-efficient infrastructures [1], [2]. However, outsourcing data processing to third-party cloud providers raises significant privacy concerns, particularly when sensitive information is involved, such as medical, financial, or personal data [3]. The access of the model to the raw data can create potential privacy risks because the data are in the cloud-shared infrastructure [4].

Conventional security mechanisms primarily protect data at rest and in transit through an encryption process. However, a decryption process is necessary when the data must be processed, falling into the initial problem of data vulnerability [5], [6]. Homomorphic Encryption (HE) has emerged as a promising solution to this challenge by enabling computations directly on encrypted data without requiring decryption [7], [8]. HE systems protect the entire data lifecycle (transmission, storage, and processing). Its correctness relies on the homomorphism concept, a structure-preserving transformation where two groups in different spaces can be mapped, e.g., polynomial rings, Galois fields, etc. Under this property, operations performed on ciphertexts correspond, after decryption, to the same operations applied to the original plaintext values.

Despite its strong security guarantees, current HE schemes impose strict limitations on the types of operations that can be executed efficiently, as they natively support only arithmetic additions and multiplications. These constraints implement complex ML models particularly challenging. In recent years, significant efforts have been devoted to integrating HE cryptosystems with Logistic Regression (LR) models. However, a fundamental obstacle remains: the standard logistic activation function is non-polynomial and relies on operations—such as exponentiation and division—that are not supported by HE cryptosystems [9]. Therefore, the efficient implementation of a cryptographically computable Sigmoid function becomes imperative for developing a privacy-preserving LR model with HE (LR-HE).

Several polynomial approximation techniques have been proposed to address this limitation in LR-HE models, including Taylor expansions, Chebyshev polynomials, least-squares fitting, and composite polynomials, etc. While these approximation methods enable HE-compatible inference, they often suffer from a trade-off between approximation accuracy and polynomial degree. Higher-degree polynomials improve fidelity but significantly increase computational cost and noise growth in HE, reducing practical feasibility.

The Cheon–Kim–Kim–Song (CKKS) scheme [10] is particularly relevant in this context, as it supports approximate arithmetic over encrypted real numbers. While CKKS uses state-of-the-art solutions to support approximations, they are still inefficient in practice, and thus, the design of efficient HE-compliant activation functions continues to be an active research area [11].

Several limitations arise in reproducing the behavior of the model using approximations in the ciphertext space [12]. The main challenge is to control the trade-off between having a non-linear transformation, which is needed by the learning algorithm, and keeping the degree of the polynomials small to make HE feasible [13], [14], [15].

To address these challenges, this paper proposes an adaptive LR-HE model enhanced with Self-Learning Activation Functions (SLAF). Instead of relying on fixed polynomial approximations of the sigmoid function, our approach introduces trainable polynomial activation functions whose coefficients are jointly optimized with the model parameters during training. This enables the model to learn task-specific and dataset-specific nonlinearities while remaining compatible with HE constraints.

By allowing activation functions to adapt to the data, our method improves both the accuracy and efficiency of LR-HE compared to static approximations. Furthermore, the proposed framework preserves end-to-end data confidentiality, making it suitable for privacy-sensitive applications in cloud environments, particularly in healthcare and other domains where secure data processing is essential.

The content of the paper is structured as follows. Section II introduces homomorphic encryption schemes and presents the primitives of the CKKS scheme. Section III describes privacy-preserving logistic regression. Section IV discusses state-of-the-art LR-HE models that polynomially approximate the cryptographically non-computable Sigmoid function. Section V introduces homomorphic self-learning activation functions. Section VI defines the experimental setup. The experimental results are presented in Section VII. Finally, we conclude in Section VIII.

II. CKKS Homomorphic Encryption Scheme

Homomorphic Encryption (HE) cryptosystems enable algebraic operations to be directly computed on encrypted data without requiring access to the secret key [7], [16]. As a result, data can be processed in encrypted form while preserving confidentiality, since both the inputs and the outputs of the computation remain encrypted. HE systems protect the entire data lifecycle (transmission, storage, and processing). Its correctness relies on the homomorphism concept, a structure-preserving transformation where two groups in different spaces can be mapped, e.g., polynomial rings, Galois fields, etc. Under this property, operations performed on ciphertexts correspond, after decryption, to the same operations applied to the original plaintext values.

Let m_a denote the message a in plaintext, sk a secret key for decryption, and pk a public key for encryption. The corresponding ciphertext c_a of m_a is generated by the encryption operation $c_a = Encrypt(pk, m_a)$. Recovering the information in an additively HE from a ciphertext c_+ is performed by the decryption operation and the sk as $m_+ = Decrypt$, where $c_+ = Add(c_a, c_b)$ contains the result of the homomorphic addition between c_a and c_b. Similarly, in a multiplicative HE scheme, the homomorphic multiplication operation produces $c_\times = Mult(c_a, c_b)$. The HE schemes obtain ciphertexts c_+ and $c_\times$, without knowing m_a and m_b. Ciphertexts c_+ and $c_\times$ cannot be computed with standard encryption without the decryption of c_a and c_b. Thus, an HE scheme is a public-key encryption scheme defined by $KeyGen$, $Encrypt$, $Decrypt$, and homomorphic evaluation operations over ciphertexts.

The Cheon–Kim–Kim–Song (CKKS) scheme is a lattice-based HE scheme whose security is based on the hardness of the Ring Learning with Errors (RLWE) problem. CKKS cryptosystem performs approximate arithmetic on encrypted real (complex) numbers, making it particularly well suited for privacy-preserving ML applications involving continuous values. Given plaintext messages m_a and m_b, the scheme allows the encrypted evaluation of approximate values of $m_a + m_b$ and $m_a \cdot m_b$ with a predefined precision.

A distinctive feature of CKKS is that the noise introduced by the RLWE construction is treated as part of the approximation error inherent to the computation. In CKKS, after selecting parameters such as the polynomial ring degree N and the ciphertext modulus q, real-valued inputs are encoded into plaintext polynomials. The sk is a randomly generated polynomial. The pk is created using sk, chosen parameters, and some randomness. The plaintext polynomial is encrypted using the pk and noise to increase security. After operations, rescaling is performed to reduce noise and preserve the ciphertext. Finally, decryption using the sk yields an approximate plaintext polynomial, which is decoded to recover the corresponding real or complex output value.

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

The distribution $\chi_{key} = HW(h)$ outputs a polynomial from R_q of $\{\pm 1\}$-coefficient having h non-zero coefficients, where HW(h) denotes the set of signed binary vectors in $\{\pm 1\}^N$ whose Hamming weight is $h \in Z_+$. χ_{enc} and χ_{err} denote discrete Gaussian distributions with some predefined standard deviation. $U(R_q)$ refers to a uniform distribution over the modulo-q residue ring R_q. Moreover, for $a = \sum_{i=0}^{N-1} a_i X^i \in R[X]/(X^N + 1)$, then $\lfloor a \rceil = \sum_{i=0}^{N-1} \lfloor a_i \rceil X^i \in R$, where $\lfloor \cdot \rceil$ returns the nearest integer of a real-number input, rounding upwards in case of a tie [10].

CKKS encodes real values using a called canonical embedding $\tau: R[X]/(X^N + 1) \to C^{N/2}$, where a plaintext vector $\vec{m} = (m_0, \ldots, m_{N/2})$ is transformed into $\tau^{-1}(\vec{m}) \in$

$R[X]/(X^N+1)$ and then rounded to an integer-coefficient polynomial using a scaling factor Δ, i.e., $\lfloor \Delta \cdot \tau^{-1}(\vec{m}) \rceil$. The parameter Δ affects the accuracy of the computation in CKKS.

Let us discuss the main primitives of the CKKS scheme:

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

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

- $\mathrm{Resc}(c) \rightarrow c'$: For a ciphertext $c \in R_{q_l}^2$, outputs $c' = \lfloor q_{l'}/q_l \rceil \cdot c (mod q_{l'})$.

In this paper, CKKS was employed as the underlying HE cryptosystem. For more detailed information and additional considerations on the CKKS, including the correctness and security analysis, refer to [10].

III. Logistic Regression

Logistic Regression (LR) is a statistical learning method for modelling the relationship between a set of input variables and a binary (dichotomous) outcome. It has been successfully applied across multiple domains, including healthcare, finance, and security-sensitive data analytics. LR models the probability of class membership through a non-linear transformation of a linear predictor using the sigmoid (logistic) function.

Consider a dataset consisting of N samples, each represented by a feature vector of dimensionality d. The input data and corresponding binary labels are denoted by $\mathrm{X} \in \mathbb{R}^{N \times d}$ and $\mathrm{Y} \in \{0,1\}^N$, respectively. Each element is represented as $\mathrm{x}^{(i)} \in \mathbb{R}^d$ with an associated label $y^{(i)} \in \{0,1\}$, for $i = 1,2, \ldots, N$.

The LR inference function is defined as

$$h_\theta(\mathrm{x}^{(i)}) = g(\theta^\top \mathrm{x}^{(i)}),$$

where θ^T is a weight vector, the element $x^{(i)}$, and $g(\cdot)$ stands for the Sigmoid function given by

$$g(z) = \frac{1}{1 + e^{-z}}$$

with the linear combination

$$z = \theta^\top \mathrm{x}^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_d x_d^{(i)}.$$

The sigmoid function maps real-valued inputs to the interval $(0, 1)$, allowing the model output to be interpreted as a probability estimate.

The efficiency of the inference in LR depends on the likelihood function expressed in terms of the parameter θ. The training phase of LR aims to determine the optimal parameter vector θ^* that maximizes the likelihood of correctly classifying the training samples. This objective is commonly achieved by minimizing the cost function error $J(\theta)$ defined as

$$J(\theta) = -\frac{1}{\mathrm{N}} \sum_{i=1}^{N} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)$$

The cost function $J(\theta)$ quantifies the discrepancy between predicted probabilities and true labels; lower values indicate improved classification performance. Importantly, for standard LR, this cost function is convex, guaranteeing convergence to a global minimum under appropriate optimization conditions.

Gradient Descent (GD) is a commonly employed first-order optimization method for minimizing $J(\theta)$. At each iteration, the parameter vector is updated in the direction of the negative gradient $-\nabla_\theta J(\theta)$. The magnitude of each update is controlled by the learning rate α, which plays a critical role in convergence behavior. Small learning rates may lead to slow convergence or stagnation, while excessively large values can cause oscillations or divergence.

Several variants of GD have been proposed in the literature, differing primarily in the update rule of θ: Batch Gradient Descent (BGD), Stochastic Gradient Descent (SGD), Momentum Gradient Descent (MGD), Nesterov Accelerated Gradient (NGD), among others.

BGD computes the gradient using the entire training dataset at each iteration. While its updates are stable and deterministic, BGD suffers from high computational and memory costs, making it impractical for large-scale datasets. SGD updates the parameters using individual samples or small subsets of the data, typically combined with random shuffling to reduce bias. This approach significantly improves convergence speed and enables online learning, but introduces gradient noise that may prevent convergence to an exact minimum. MGD incorporates a momentum term that accumulates past gradient updates, reducing oscillations and accelerating convergence along

consistent descent directions. This behaviour is often likened to the acceleration of a ball rolling down a slope. Finally, NGD considers an approximation of the future position in the search to improve the performance of MGD. Its update rule uses the momentum term, where the gradient is missing a full update to change θ; it can significantly increase the performance of MGD. Following the analogy of the ball in MGD, NGD provides a ball with a notion of the direction that can slow down before the hill slopes up again.

Adaptive learning-rate methods such as Adagrad, Adadelta, RMSprop, Adam, AdaMax, and Nadam further enhance optimization by dynamically adjusting α based on historical gradient information. While effective in many scenarios, these methods introduce additional computational complexity and hyperparameters. Second-order optimization approaches, such as the Newton–Raphson method, leverage curvature information by incorporating the Hessian matrix. Although these methods can achieve rapid convergence, their computational and memory requirements render them impractical for high-dimensional datasets and large-scale learning problems.

Once training is complete, the optimized parameter vector θ^* is used to classify unseen samples. Given a new input $\mathrm{x}' \in \mathbb{R}^{d+1}$, the predicted label y' is obtained by thresholding the output probability

$$y' = \begin{cases} 1, & \text{if } h_{\theta^*}(\mathrm{x}') \geq \tau, \\ 0, & \text{otherwise,} \end{cases}$$

where the threshold $0 < \tau < 1$ is typically equal to 0.5.

Despite its favourable properties—such as interpretability, convex optimization, and competitive performance—LR presents notable challenges when deployed in privacy-sensitive settings. In particular, both training and inference processes traditionally require access to plaintext data and model parameters, which raises significant concerns in untrusted or shared environments. These limitations motivate the integration of LR with privacy-preserving techniques, such as lattice-based HE schemes.

IV. Related Work

High-demand solutions such as LR-HE modelling require efficient operations beyond HE additions and multiplication [17], [18]. Moreover, an important challenge toward deploying HE in cognitive models is overcoming the high computational costs associated with these cryptosystems. Computing over ciphertexts with state-of-the-art cryptosystems such as the CKKS scheme represents a slowdown of 4-6 orders of magnitude compared to performing the same operations on unencrypted data [19]. While several methods have been proposed in the literature to accelerate the homomorphic processing of these solutions, they remain inefficient in practice. Therefore, designing efficient cryptographically compatible solutions to optimize privacy-preserving models remains an open challenge.

Naehrig et al. [20] investigate privacy-preserving LR using a Ring-LWE-based somewhat HE scheme and efficient message encoding techniques to support polynomial evaluations over encrypted data. While their approach demonstrates the feasibility of encrypted statistical analysis and regression-style computations, it does not provide an HE-native approximation of the logistic function; instead, non-linear operations such as exponentiation and division are deferred outside the encrypted domain or handled implicitly, which limits the ability to perform fully encrypted end-to-end LR inference.

Aono et al. [21] propose a scalable LR-HE model and show that encrypted training can be achieved using only additive schemes by reformulating the learning objective. Specifically, they derive a homomorphism-aware LR solution by approximating the log-likelihood terms with a low-degree polynomial approximation, enabling evaluation of the approximate cost function using ciphertext additions and scalar multiplications.

Kim et al. [22] propose a practical homomorphic framework for LR using the CKKS scheme, where both training and inference are performed over encrypted data. To enable non-linear computations, they approximate the sigmoid function with global least-squares polynomials of degrees 3, 5, and 7 over a fixed interval, allowing encrypted GD to be evaluated using only additions and multiplications.

Patra and Suresh [23] propose BLAZE, a privacy-preserving ML framework for secure outsourced computation that supports LR and Neural Network (NN) models under a three-server threat model with one potentially corrupted party. Their design substantially improves round complexity and communication efficiency over both wide-area and local-area networks, outperforming existing secure ML protocols for a range of learning tasks.

De Cock et al. [24] introduced secure training of LR in a multi-party computation setting, where data are distributed across multiple parties. They introduce a protocol that supports an efficient approximation of the sigmoid function together with cryptographic optimizations that significantly reduce computation and communication overhead. The solution provides an efficient training time of 26.90 seconds in local area networks. The two-party computation model can provide a fast and secure genome analysis.

An LR model can be viewed as a single-layer NN model with a single output neuron and a nonlinear activation function. Consequently, the challenge of implementing LR-HE solutions is a special case of the broader problem of supporting NN inference and training over encrypted data. In particular, approximating the logistic activation in LR-HE is analogous to approximating activation functions in NN-HE models, where non-linear operations must be replaced with cryptographically computable alternatives. This connection has motivated a substantial body of work on privacy-preserving NN-HE models [16], [25].

CryptoNets, introduced by Dowlin et al. [26], represents the first model that addresses the challenge of achieving a blind, non-interactive classification. It uses the leveled YASHE [27] for inputs and propagates signals across the network homomorphically. Although it enabled blind classification, its reliance on a square activation function, high computational cost, and security limitations of YASHE restricted both accuracy

and practicality. Subsequent work sought to overcome these constraints. Chabanne et al. [28] improved CryptoNets by introducing polynomial approximations of ReLU and batch normalization, while Chou et al. [29] leveraged pruning and quantization to accelerate encrypted inference under the BFV scheme.

Bourse et al. [30] proposed Discretized Neural Network (DiNN) models for inference over encrypted data, which achieve linear complexity in network size but require expensive bootstrapping under the Fully Homomorphic Encryption scheme over the Torus (TFHE), resulting in high overhead and reduced accuracy. Sanyal et al. [31] extended this idea to Binary NNs, enabling arithmetic through binary gate composition, but at the cost of extremely high latency, with inference times reaching several hours per input. Van Elsloo et al. [32] introduced SEALion, an extensible CryptoNets-based framework that automates encryption parameter selection to improve encrypted inference efficiency.

CryptoDL [33] replaces standard nonlinearities with HE-friendly low-degree polynomials, while Liao et al. [34] further extend this approach by approximating Tanh, ReLU, and Swish for encrypted sensor-data inference. Brutzkus et al. [35] proposed Low-Latency CryptoNets (Lo-La), which pack entire layers into single ciphertexts to reduce memory and latency, although they remain constrained by ciphertext dimension. nGraph-HE [36] and E2DM [37] introduce HE-aware compilation and SIMD-based encrypted matrix arithmetic, enabling large-scale encrypted inference, including fast batched evaluation on MNIST [38].

Hardware-accelerated and high-capacity models have also been explored. Badawi et al. [12] presented an efficient GPU-based BFV implementation for NN-HE solutions, significantly accelerating the classification process while maintaining accuracy; it classifies MNIST in 2% of the time CryptoNets takes. Lee et al. [39] demonstrated deep ResNet-20 inference using RNS-CKKS with bootstrapping, but with inference times of several hours per image due to the cryptographic overhead and high-degree polynomial approximations. Pulido-Gaytan and Tchernykh [40] proposed non-interactive privacy-preserving SLAF-based CNN-HE models to classify CKKS-encrypted data with improved accuracy and performance. The authors show that a self-learning mechanism can achieve the same accuracy as a non-polynomial ReLU over non-homomorphic CNNs and increase accuracy and higher performance than the state-of-the-art CNN-HE CryptoNets. This line of work provides strong evidence that learning the activations themselves, rather than relying on fixed polynomial approximations, is a powerful strategy for improving both accuracy and efficiency in encrypted ML models.

V. Self-Learning Activation Functions

In this section, we present the SLAF technique to design a privacy-preserving LR-HE model. We approximate the non-linear activation function by a polynomial with trainable coefficients as

$$\ddot{f}_k = a_0^k + a_1^k x + a_2^k x^2 + \cdots + a_n^k x^n$$

where $a_0^k, a_1^k, \ldots, a_n^k$ denote the trainable coefficients of the polynomial $\ddot{f}_k$.

The fundamental objective of the SLAF training process is to learn an effective mapping between the input and output spaces by jointly optimizing LR weights and the coefficients of polynomial activation functions. In contrast to conventional LR models with fixed nonlinearities, this approach allows the model to adapt both its synaptic weights and its activation behaviour during training. As a result, LR models equipped with SLAFs can capture richer representations from training data.

Beyond adjusting synaptic weights, the training phase explicitly optimizes the coefficients of the SLAF polynomials. Consequently, the LR-HE-SLAF activations become tailored to the specific learning task, the chosen hyperparameters, and the characteristics of the dataset. This joint optimization enables the model to learn problem-specific nonlinearities while remaining compatible with the arithmetic constraints imposed by HE cryptosystems.

Given that SLAF can approximate any continuous activation function f if its input domain is bounded to a desired error as a function of the SLAF degree [40], [41], we consider SLAF with $n+1$ trainable polynomial coefficients $a_0, a_1, \ldots, a_n$ initialized by zero, allowing a search from the "zero point" of the solution space.

By allowing activation functions to be learned rather than fixed, SLAF-based LR-HE models not only uncover underlying relationships in the data but also identify activation behaviors that are inherently suited to a given dataset and computational constraint. This property is particularly valuable in privacy-preserving settings, where the form of nonlinear operations must be carefully designed to balance expressiveness, efficiency, and cryptographic compatibility.

VI. Experimental Setup

The HE schemes, homomorphic operations, and NN-HE models are implemented using PyTorch [42] and the open-source Simple Encrypted Arithmetic Library (SEAL) v3.5.6 [43] through the Python TenSEAL library [44]. The experimental evaluation is performed on a server Express x3650 M4 with Intel(R) Xeon(R) CPU E5-2650v2 95W at 2.6GHz, 64 GB. The server OS is Ubuntu 18.04.6 of 64 bits.

The experiments presented in this paper were performed during a prior phase of a continuous, long-term investigation of privacy-preserving learning under HE. These results have not been previously published and provide a reliable, consistent, and reproducible evaluation of the proposed approach.

This section describes the evaluation methodology, dataset characteristics, and the experimental setup.

A. Security settings

The ciphertext size, scheme performance, multiplicative depth, and security level depend on the security parameter settings. We adopt the security settings specified in the HE standard [45]. Table 1 summarizes the parameters used for the CKKS scheme, where λ denotes the security level in bits, N is the polynomial modulus degree, Δ represents the scaling factor used for approximate arithmetic, L is the supported

multiplicative depth, and q denotes the ciphertext modulus. The security level $\lambda = 128$ bits guarantees that an adversary needs to perform 2^{128} elementary operations to break the scheme with a probability one. A detailed description of these parameters is provided in Section II.

TABLE I. SECURITY SETTINGS FOR CKKS SCHEME

Parameter	Security settings
λ	128
N	2^{13}
Δ	2^{21}
L	8
q	$[40, 21, \ldots, 21, 40]$

B. Dataset

An evaluation on real-world data is essential to assess the practical performance of the proposed strategies. To this end, we employ the Heart Disease Prediction dataset, a benchmark dataset widely used in the literature. It includes patients' information along with a 10-year risk of future Coronary Heart Disease (CHD) as a label. The goal is to build a model that can predict this 10-year CHD risk based on patient clinical information and demographic attributes.

This dataset contains over 4000 patient records with clinical, behavioral, and demographic features, including variables such as age, sex, smoking status, blood pressure, cholesterol levels, and other health indicators that are known risk factors for heart disease. Each record is paired with a binary label indicating whether a patient is expected to develop CHD within ten years: 1 = risk present, 0 = no risk.

Continuous input features are scaled to the interval [0,1] using min–max normalization

$$x_{nv} = \frac{x - \min(x)}{\max(x) - \min(x)},$$

where x is the original value, and x_{nv} is the normalized value. Feature scaling is critical for both numerical stability in the training process and for controlling noise growth in HE computations.

To ensure a robust evaluation of the LR-HE model generalization, we adopt 5-fold Cross-Validation (5CV). This protocol randomly partitions the dataset into five equally sized folds; in each iteration, four folds are used for training and the remaining fold for the testing task. The 5CV strategy is standard practice in the literature. Table 2 summarizes the main characteristics of the dataset used, including the total number of samples, the number of input features, and the size of the training and testing subsets under the 5-fold cross-validation protocol.

TABLE II. DATASET CHARACTERISTICS

Dataset	N	Features	N-Training	N-Testing
Heart Disease Prediction	4,240	15	3,392	848

C. Evaluation method

The performance of a classifier is determined by the number of correct and incorrect predictions for each class. A Confusion Matrix (CM) is commonly used to summarize the correspondence between true and predicted labels over a set of examples. From the CM, several meaningful performance metrics can be derived, including Accuracy (A), Precision (P), Recall (R), and Specificity (S). Accuracy measures the overall proportion of correct predictions. Precision measures the proportion of predicted positive instances that are correctly classified. Recall (also known as sensitivity) measures the proportion of actual positive instances that are correctly identified, while Specificity measures the proportion of actual negative instances that are correctly classified. These metrics are defined as follows

$$A = \frac{T_p + T_n}{T_p + T_n + F_p + F_n},$$

$$P = \frac{T_p}{T_p + F_p},$$

$$R = \frac{T_p}{T_p + F_n},$$

$$S = \frac{T_n}{T_n + F_p},$$

where T_p and T_n denote the number of correctly classified positive and negative samples, respectively, and F_p and F_n represent the number of incorrectly classified positive and negative samples.

The choice of classification threshold can significantly influence the values of A, P, R, and S. Therefore, a threshold-independent metric is desirable for comparing different models. The Receiver Operating Characteristic (ROC) curve provides such a tool by illustrating the trade-off between the True Positive Rate (TPR), equivalent to Recall, and the False Positive Rate (FPR), defined as $FPR = F_p/(F_p + T_n)$, across varying decision thresholds applied to the classifier outputs. Each point on the ROC curve corresponds to a specific threshold value.

Direct visual comparison of ROC curves can be challenging. Consequently, the Area Under the ROC Curve (AUC) is commonly used as a scalar summary that quantifies the overall discriminative ability of a classifier. The AUC corresponds to the probability that the classifier ranks a randomly selected positive instance higher than a randomly selected negative instance. Together, A and AUC provide complementary criteria for evaluating and comparing the performance of the proposed methods.

VII. EXPERIMENTAL ANALYSIS

This section presents the experimental results of the proposed privacy-preserving LR-HE model evaluation on encrypted data for heart disease prediction—the assessment of

the privacy-preserving solution centers on its accuracy and latency.

A. LR-HE Training

Based on the security parameters reported in Table I, we employ a ciphertext modulus chain of eight co-primes. While the encrypted training pipeline has a multiplicative depth of six—one level for the dot product, two for the polynomial approximation, and three for backpropagation—the CKKS scheme incurs additional level consumption due to rescaling and scale alignment procedures. Under this 128-bit security setup, we use a scaling factor of 21 bits and a polynomial modulus degree of $N = 8192$.

A security level of $\lambda = 128$ bits for our configuration guarantees that an adversary needs to perform 2^{128} elementary operations to break the cryptosystem with a probability of one [46], [47]. The polynomial modulo degree N allows them to support polynomials of degree at most $2^{13} - 1$. A larger N allows for a larger ciphertext coefficient q to be used without decreasing λ, but makes ciphertext sizes bigger and homomorphic operations slower.

Since the logistic function cannot be evaluated directly over encrypted data, it must be replaced by a polynomial approximation, where lower degrees are preferred to minimize the multiplicative depth and the associated HE overhead. The main challenge is to control the trade-off between having a non-linear transformation, which is needed by the learning algorithm, and keeping the degree of the polynomials small to make HE feasible [13], [14], [15]

We consider two approaches. First, we use a fixed degree-3 polynomial approximation of the sigmoid over the interval $[-5,5]$, as proposed in Chen et al. [48], which provides a good trade-off between approximation accuracy and computational depth. Second, we adopt the SLAF method, in which the coefficients of a low-degree polynomial are trained jointly with the model parameters, allowing each activation to adapt to the data and network structure. SLAFs can approximate continuous activation functions with arbitrarily small error while maintaining low polynomial degree and HE efficiency. We evaluate both approaches in our LR-HE framework and compare their impact on prediction accuracy and encrypted computation latency.

We analyze the distribution of the pre-activation term $x \cdot w + b$ in both the plaintext and encrypted spaces to ensure that it remains within the approximated interval. Since synaptic weights evolve during the training process, we apply L_2 regularization in order to constrain their magnitude and prevent the pre-activation values from drifting outside this range.

Figure 1 shows the distribution of the pre-activation inputs in both the plaintext and ciphertext spaces, showing that the vast majority of values remain within the target interval, preserving numerical stability and approximation accuracy during encrypted training.

The LR model is trained using Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and momentum 0.9 for 100 epochs. All input features are standardized to zero mean and unit variance prior to training, which stabilizes optimization and ensures that the pre-activation values remain within the range required by the polynomial approximation of sigmoid and SLAF approximations.

B. LR-HE Evaluation

We evaluated the performance of the privacy-preserving LR-HE solution. Table III presents the LR-HE model latency (Lat) and accuracy (Acc) for ciphertext inputs. Latency is the time required to process a single encrypted request. On the other hand, Table IV provides a more detailed characterization of the model performance by reporting A, P, R, S, and AUC metrics (see Section VI.C), enabling a comprehensive comparison between static polynomial approximations and the proposed SLAF-based model under HE.

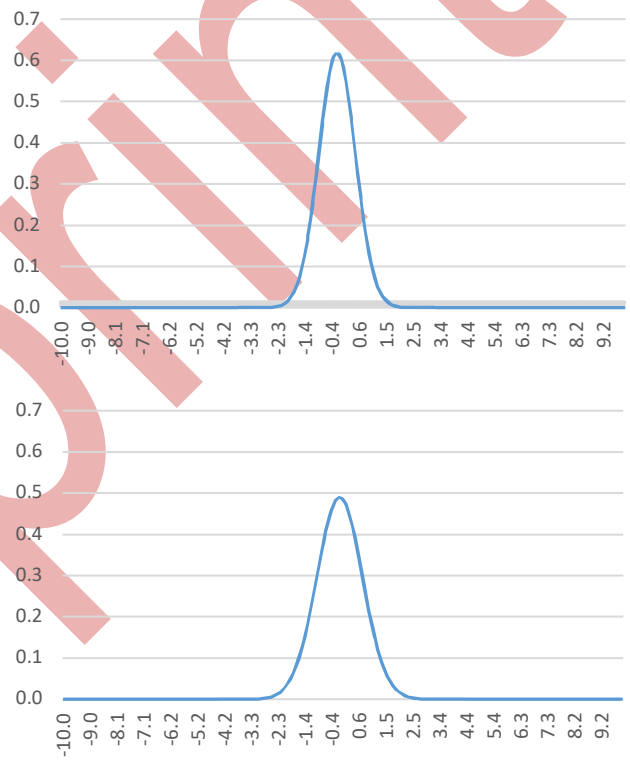

Fig. 1. Distribution of the pre-activation values for both the plaintext (top) and encrypted (bottom) domains.

TABLE III. PERFORMANCE OF LR-HE

Model	Polynomial Approximation	Acc (%)	Lat (ms)
LR-HE	Minimax	69.37	156
	SLAF	76.58	129

We can see that the LR-HE implementation with a three-degree polynomial SLAF improves accuracy concerning the static minimax composition approach. By learning the polynomial activation during training, SLAF improves accuracy from 69.37% to 76.58% while also reducing encrypted latency from 156 ms to 129 ms. These results highlight the efficiency of LR-HE in terms of processing time without compromising

accuracy, suggesting promising potential for practical applications.

TABLE IV. ACCURACY (A), PRECISION (P), RECALL (R), SPECIFICITY (S), AND AUC OF PRIVACY-PRESERVING LR-HE MODEL

Model	Polynomial	A	P	R	S	AUC
LR-HE	Minimax	69.37	71.91	59.81	78.26	75.22
	SLAF	76.58	77.57	74.77	78.38	83.40

According to Table IV, SLAF-based LR-HE model consistently outperforms the minimax baseline across all evaluation metrics, increasing P from 71.91% to 77.57% and R from 59.81% to 74.77%, while maintaining a similar level of specificity. The increase in Recall is particularly important in medical decision-support scenarios, as it corresponds to a substantial reduction in false negatives. Moreover, AUC improves from 75.22% to 83.40%, indicating a markedly stronger ability to discriminate between positive and negative cases across different decision thresholds.

These results demonstrate that adaptive, trainable polynomial activations provide a more effective approximation of the logistic function under homomorphic constraints than fixed approximations, enabling a better accuracy–latency trade-off for privacy-preserving LR-HE implementations.

VIII. CONCLUSION

Building on the principle of protecting computation and not just data, this paper focuses on the design of privacy-preserving LR models for classifying encrypted information using lattice-based HE cryptosystems that enable secure data processing in remote cloud environments. The homomorphic processing of cognitive models requires operations not supported by HE schemes, so efforts focused on designing cryptographically compatible non-linear functions capable of operating over ciphertexts.

In this paper, we propose an adaptive LR-HE model enhanced with SLAF. Instead of relying on fixed polynomial approximations of the sigmoid function, our approach introduces trainable polynomial activation functions whose coefficients are jointly optimized with the model parameters during training. Experimental analysis on the Heart Disease Prediction dataset shows that the proposed SLAF-based LR-HE model reduces latency compared to state-of-the-art approximation solutions without compromising security and improving accuracy.

In future work, we will conduct a systematic review of state-of-the-art hardware acceleration techniques to mitigate the computational overhead inherent to HE and to improve the practicality of LR-HE in real-world deployments. These include highly parallel CPUs, graphic processing units (GPUs), field-programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), etc. In addition, we plan to implement and evaluate the proposed models in realistic pilot studies, such as medical image classification and privacy-preserving smart transportation forecasting, to demonstrate their effectiveness in privacy-sensitive environments.

ACKNOWLEDGMENT

The research conducted in this publication was funded by the European Commission under grant 101186291 (SMARCO: SMARt COmmunities Skills Development in Europe).

REFERENCES

[1] S. Kaushik and C. Gandhi, "Capability based outsourced data access control with assured file deletion and efficient revocation with trust factor in cloud computing," International Journal of Cloud Applications and Computing, vol. 10, no. 1, pp. 64–84, 2020, doi: 10.4018/IJCAC.2020010105.

[2] T. Hai, J. Zhou, Y. Lu, D. N. A. Jawawi, D. Wang, S. Selvarajan, H. Manoharan, and E. Ibeke, "An archetypal determination of mobile cloud computing for emergency applications using decision tree algorithm," Journal of Cloud Computing, vol. 12, no. 1, p. 73, 2023, doi: 10.1186/s13677-023-00449-z.

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

[4] S. Shitharth, K. M. Prasad, K. Sangeetha, P. R. Kshirsagar, T. S. Babu, and H. H. Alhelou, "An enriched RPCO-BCNN mechanism for attack detection and classification in SCADA systems," IEEE Access, vol. 9, pp. 156297–156312, 2021, doi: 10.1109/ACCESS.2021.3129053.

[5] A. Tchernykh, M. Babenko, V. Kuchukov, V. Miranda-López, A. Avetisyan, R. Rivera-Rodriguez, and G. Radchenko, "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, doi: 10.1109/IPDPSW.2019.00099.

[6] Q. Zheng, X. Wang, M. Khurram Khan, W. Zhang, B. B. Gupta, and W. Guo, "A lightweight authenticated encryption scheme based on chaotic SCML for railway cloud service," IEEE Access, vol. 6, pp. 711–722, 2018, doi: 10.1109/ACCESS.2017.2775038.

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

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

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

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

[11] M. Babenko, A. Tchernykh, E. Golimblevskaia, B. Pulido-Gaytan, and A. Avetisyan, "Homomorphic comparison methods: Technologies, challenges, and opportunities," in 2020 International Conference Engineering and Telecommunication (En&T), 2020, pp. 1–5, doi: 10.1109/EnT50437.2020.9431252.

[12] A. Al Badawi, C. Jin, J. Lin, C. F. Mun, S. J. Jie, B. H. M. Tan, X. Nan, K. M. M. Aung, and V. R. Chandrasekhar, "Towards the AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs," IEEE Trans. Emerg. Top. Comput., vol. 9, no. 3, pp. 1330-1343, 2020, doi: 10.1109/TETC.2020.3014636.

[13] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, "Numerical method for comparison on homomorphically encrypted numbers," in Advances in Cryptology, 2019, pp. 415–445, doi: 10.1007/978-3-030-34621-8_15.

[14] E. Lee, J.-W. Lee, J.-S. No, and Y.-S. Kim, "Minimax approximation of sign function by composite polynomial for homomorphic comparison,"

IEEE Trans Dependable Secure Comput, vol. 19, no. 6, pp. 3711–3727, 2022, doi: 10.1109/TDSC.2021.3105111.

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

[16] C. Gentry and S. Halevi, "Implementing Gentry's fully homomorphic encryption scheme," in Advances in Cryptology, 2011, pp. 129–148, doi: 10.1007/978-3-642-20465-4_9.

[17] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in Network and Distributed System Security Symposium, 2015. doi: 10.14722/ndss.2015.23241.

[18] C. Gentry, "Computing arbitrary functions of encrypted data," Communications of the ACM, vol. 53, no. 3, pp. 97–105, 2010, doi: 10.1145/1666420.1666444.

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

[20] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in 3rd ACM Workshop on Cloud Computing Security Workshop, 2011, pp. 113–124. doi: 10.1145/2046660.2046682.

[21] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang, "Scalable and secure logistic regression via homomorphic encryption," in 6th ACM Conference on Data and Application Security and Privacy, 2016, pp. 142–144, doi: 10.1145/2857705.2857731.

[22] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," BMC Med Genomics, vol. 11, p. 83, 2018, doi: 10.1186/s12920-018-0401-7.

[23] A. Patra and A. Suresh, "BLAZE: Blazing fast privacy-preserving machine learning," arXiv preprint arXiv:2005.09042, 2020.

[24] M. De Cock, R. Dowsley, A. C. A. Nascimento, D. Railsback, J. Shen, and A. Todoki, "High performance logistic regression for privacy-preserving genome analysis," BMC Med Genomics, vol. 14, no. 1, p. 23, 2021, doi: 10.1186/s12920-020-00869-9.

[25] 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.

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

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

[28] 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.

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

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

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

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

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

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

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

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

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

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

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

[40] B. Pulido-Gaytan and A. Tchernykh, "Self-learning activation functions to increase accuracy of privacy-preserving convolutional neural networks with homomorphic encryption," PLoS ONE, vol. 19, no. 7, p. e0306420, doi: 10.1371/journal.pone.0306420.

[41] L. Hou, D. Samaras, T. M. Kurc, Y. Gao, and J. H. Saltz, "ConvNets with smooth adaptive activation functions for regression," in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR, 2017, pp. 430–439.

[42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, et al., "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems (NeurIPS), 2019, p. 8026–8037.

[43] H. Chen, K. Laine, and R. Player, "Simple Encrypted Arithmetic Library," 2019.

[44] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "TenSEAL: A library for encrypted tensor operations using homomorphic encryption," arXiv:2104.03152. 2021.

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

[46] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from Ring-LWE and security for key dependent messages," in Advances in Cryptology, 2011, pp. 505–524, doi: 10.1007/978-3-642-22792-9_29.

[47] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) Fully homomorphic encryption without bootstrapping," ACM Trans. Comput. Theory, vol. 6, no. 3, pp. 1-36, 2014, doi: 10.1145/2633600.

[48] H. Chen, R. Gilad-Bachrach, K. Han, Z. Huang, A. Jalali, K. Laine, and K. Lauter, "Logistic regression over encrypted data from fully homomorphic encryption," BMC Med Genomics, vol. 11, p. 81, 2018, doi: 10.1186/s12920-018-0397-z.