PRISM: Privacy-preserving Inference System with Homomorphic Encryption and Modular Activation

Zeinab Elkhatib[1], Ali Sekmen [1], Kamrul Hasan[1]
[1] Tennessee State University, Nashville, TN, USA
Email: {*zelkhati, asekmen, mhasan1*}@tnstate.edu

***Abstract*—With the rapid advancements in machine learning, models have become increasingly capable of learning and making predictions in various industries. However, deploying these models in critical infrastructures presents a major challenge, as concerns about data privacy prevent unrestricted data sharing. Homomorphic encryption (HE) offers a solution by enabling computations on encrypted data, but it remains incompatible with machine learning models like convolutional neural networks (CNNs), due to their reliance on non-linear activation functions. To bridge this gap, this work proposes an optimized framework that replaces standard non-linear functions with homomorphically compatible approximations, ensuring secure computations while minimizing computational overhead. The proposed approach restructures the CNN architecture and introduces an efficient activation function approximation method to mitigate the performance trade-offs introduced by encryption. Experimental results using the CIFAR-10 dataset validate the effectiveness of the proposed framework with an accuracy of 94.4% with 2.4s per inference, demonstrating its ability to preserve accuracy while significantly reducing computational complexity in encrypted environments.**

I. Introduction

Neural networks have demonstrated remarkable predictive performance and delivered innovative solutions across a broad spectrum of domains. However, when deploying these models in sensitive areas such as healthcare and critical infrastructures, it is imperative that they comply with strict data protection regulations, including HIPAA [1] and GDPR [2]. A promising strategy to address these challenges is to use publicly available data for training while employing homomorphic encryption during inference. Training on non-sensitive, public datasets minimizes the risks associated with data alteration or exposure during model development, allowing us to focus our security efforts on the inference phase [3]. This approach ensures that sensitive data remains encrypted throughout the computational process, thereby providing a robust and compliant framework for real-world deployment.

In the inference phase, applying Homomorphic encryption on the confidential data offers a powerful means to preserve data confidentiality by enabling computations to be performed directly on encrypted data. In this context, the CKKS fully homomorphic encryption scheme, named after its developers Cheon, Kim, Kim, and Song, is particularly attractive. CKKS is specifically designed for approximate arithmetic on ciphertexts, making it feasible to perform neural network inferencing without exposing the underlying sensitive information [4]. Nonetheless, integrating CKKS with convolutional neural networks, CNNs, presents significant challenges. A primary obstacle is the reliance of CNNs on non-linear activation functions, such as ReLU, Sigmoid, Tanh, and Swish, which are not directly compatible with the algebraic operations supported by CKKS [5]. To bridge this gap, researchers have developed polynomial approximations of these activation functions. These techniques can be broadly categorized into two groups. The first category utilizes lower-degree polynomials with a fixed number of multiplications, thereby prioritizing computational efficiency at the cost of reduced model accuracy [6], [7], [8]. In contrast, the second category employs higher-degree approximations in conjunction with bootstrapping to mitigate the noise introduced in ciphertexts, particularly in deeper network architectures. Although this approach can achieve higher inference accuracy, it comes with significant computational overhead and increased resource consumption [9]. Moreover, the challenge extends beyond activation functions. Other components of the CNN architecture, including convolution and pooling operations, also require adjustments to operate efficiently under the constraints of CKKS [7]. These modifications are necessary to ensure that all parts of the network are compatible with encrypted computation. The trade-off between computational efficiency and inference accuracy raises a critical question: Is it possible to design a convolutional neural network architecture with a simplified activation function that achieves high accuracy while minimizing computational time and resource usage? In this work, we address this question by proposing a novel design that carefully balances these competing factors. Our approach leverages a low-degree polynomial approximation for activation functions and modifies the CNN architecture to better suit encrypted computations. Furthermore, we focus exclusively on secure inference using homomorphic encryption while relying on publicly available datasets for training. This strategy offers a practical and efficient solution for applications in critical infrastructures, where data confidentiality, regulatory compliance, and real-time performance are essential.

The rest of this paper is organized as follows: Section II reviews related work; Section III presents the methodology; Section IV discusses the experimental results; and Section V concludes the study.

II. Related Work

In this section, previous works can be seen to have developed polynomial approximations and adjusted the Convolutional Neural Network (CNN) architecture to become compatible with handling images in the form of ciphertext. The results of each

work can be seen in Table I and Table II representing the MNIST and CIFAR-10 datasets, respectively. It can be seen that for MNIST, works [6], [7], [10], have similar accuracies and inference per image time. And even when increasing the degree of the polynomial [10], the accuracy stays relatively similar to the rest. This is because the MNIST dataset does not have too many features. Because of the limited number of features, using a shallow CNN architecture, employing a simple polynomial approximation, such as using the square function as in [7], [6] can yield surprisingly good results. This is because a shallow network, by design, minimizes the number of layers and operations, thereby reducing both computational overhead and cumulative noise in homomorphic encryption. As a result, even basic approximations are sufficient to capture the necessary nonlinearities for accurate inference while keeping the evaluation process efficient and within the noise budget. In homomorphic encryption for CNNs on datasets like CIFAR 10, striking the right balance between network depth and the degree of polynomial approximation is crucial. CIFAR 10 images,32×32×3, [11] are complex and require an architecture capable of extracting detailed, hierarchical features. A shallow CNN with only a few layers may reduce computational costs, and a simple low-degree polynomial (such as the square function) can offer a computationally efficient approximation. However, this combination, while efficient, tends to yield lower accuracy (for instance, 77.59% as shown in Table 2 [6]) because the network lacks the depth to capture all the nuances present in the data. Increasing the polynomial degree to between 15 and 27 [3] does improve accuracy, yet it still may not match the performance of more sophisticated CNN designs like those in [5], which achieved around 90%+ accuracy using a lower degree polynomial by employing a more effective architecture. On the flip side, deeper CNN architectures introduce a higher number of homomorphic operations, which in turn accumulates noise in the ciphertexts. When this noise grows too high, decryption fails unless a process called bootstrapping is applied. Bootstrapping "refreshes" the ciphertexts by reducing accumulated noise, but it is extremely time-consuming, sometimes extending inference times to as long as three hours per image, as observed in [12]. Therefore, while a shallow network with a simple polynomial is computationally attractive, it often cannot extract the rich features needed from CIFAR 10, and a deeper network, despite its potential for higher accuracy, may become impractical due to the heavy computational burden and the need for frequent bootstrapping. The key challenge is finding a middle ground that offers sufficient expressive power without incurring prohibitive computational delays.

TABLE I
RESULTS OF PREVIOUS WORKS ON MNIST DATASET

Author	Results: MNIST			
	Activation	Accuracy	Time	Degree
Chabanne et al. [7].	Square	99.3%	2.58 ms	Degree 2
Badawi et al. [6]	Square	99.0%	2.75 ms	Degree 2
Srinath et al. [5]	Softplus	99.59%	2.50 ms	Degree 2
Takumi et al. [10]	Swish	99.29%	21.15s	Degree 5
Hesamifard et al. [13]	ReLU	99.25%	2.55 ms	Degree 3

TABLE II
RESULTS OF PREVIOUS WORKS ON CIFAR-10 DATASET

Author	Results: CIFAR-10			
	Activation	Accuracy	Time	Degree
Proposed Method	Softplus	89.65%	17.38s	Degree 4
Badawi et al. [6]	Square	77.59%	304.43s	Degree 2
Srinath et al. [5]	Softplus	90.37%	–	Degree 7
Takumi et al. [10]	ReLU	81.06%	1555.5s	Degree 4
Junghyun et al. [3]	ReLU	87.9%	2,892s	Degree 15–27
Joon-Woo et al. [12]	ReLU	92.43%	3hr	Degree 16
Chou et al. [8]	Swish	75.99%	22,372s	Degree 2

III. METHODOLOGY

To improve the classification accuracy and computational efficiency of convolutional neural networks (CNNs) evaluated under homomorphic encryption (HE), we propose a complete methodology for training and deploying encrypted-compatible models. The primary constraint in encrypted inference include the need to replace the non-linear activation function with its polynomial equivalent. This is done to be compatible with encypted data and to minimize multiplicative depth in order to preserve ciphertext integrity and reduce evaluation time. To address these challenges, we approximate the Softplus function using a low-degree polynomial generated through weighted minimax optimization using Powell's method. Next, we applied batch normalization (BN) to constrain input distributions and improve approximation accuracy. Furthermore, we modified the CNN architecture to eliminate non-polynomial operations while preserving the model's depth. Finally, we train models using a two-phase procedure which allows the model to learn the weights of the model and implement them during the inference stage. The experiments were conducted on a system equipped with an Intel® Core™ i9-14900HX CPU (24 cores, 32 threads, base clock 2.20 GHz) and 31.71 GB of RAM, with evaluation focused on classification accuracy and inference time. The entire model can be visually seen in the figure below.

A. Selection of Activation Function

In this work, we propose a HE- friendly activation function by using the Softplus function. It is defined as:

$$\text{Softplus}(x) = \log\left(1 + e^{x}\right) \tag{1}$$

and is approximated into a degree-4 polynomial in the form below:

$$f(x) = Ax^4 + Bx^3 + Cx^2 + Dx + E \tag{2}$$

.

Softplus was selected because it exhibits a gradual transition and produces smoother curves, making it well-suited for approximation with a lower-degree polynomial. This is particularly important for homomorphic encryption, where lower-degree polynomial approximations reduce the number of multiplicative operations, thereby limiting the multiplicative depth. A lower multiplicative depth is critical because it minimizes

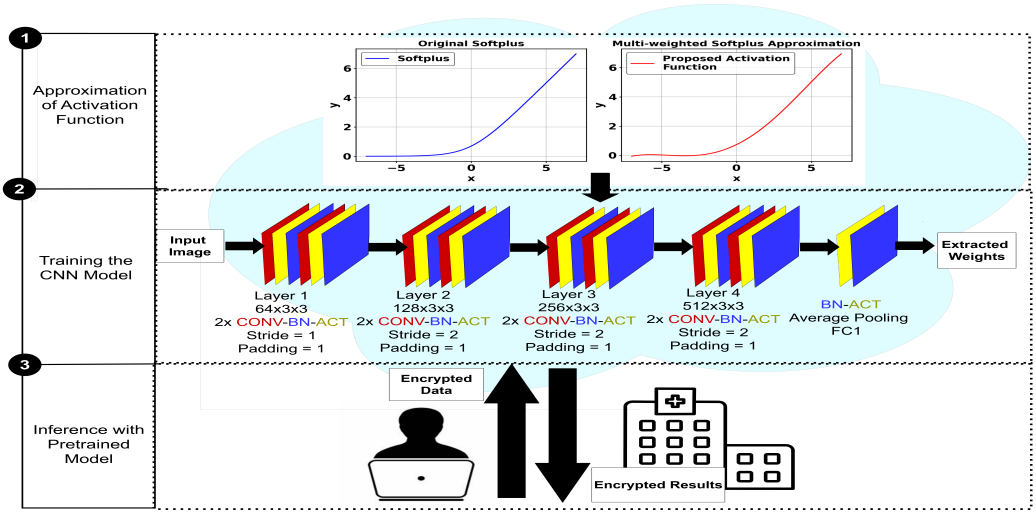

Fig. 1. Complete Model Described

noise accumulation, which can otherwise degrade evaluation accuracy.

Furthermore, although ReLU and Swish are popular activation functions and often perform well in CNNs on the CIFAR dataset, our approximation experiments show that both produce higher maximum approximation errors than Softplus, as shown in Table III, when subject to certain constraints.

TABLE III
POLYNOMIAL APPROXIMATION DETAILS AND MAX ERRORS FOR DIFFERENT ACTIVATION FUNCTIONS.

Activation Function Approximated	Degree	Range	Max Error
Softplus	4	[-7, 7]	0.067
ReLU	4	[-7, 7]	0.331
Swish	4	[-7, 7]	0.18

This is expected, especially for ReLU, whose sharp, non-smooth shape makes it harder to approximate with a low-degree polynomial. Swish also shows a larger gap than Softplus when comparing its original form and its polynomial approximation. Because of this, Softplus is often a more practical choice for approximation, even if its raw performance in plaintext CNNs is lower than that of ReLU or Swish. To fairly compare these functions, we evaluated the quality of each approximation through a three-step process: an initial weighted least squares fit, Powell's method minimax optimization, and a final maximum error measurement. For the Softplus activation, this process yielded a degree-4 polynomial where the optimized coefficients are as follows: A = -0.00068481, B = -1.5983 $\times 10^{-17}, C = 0.0887234775, D = 0.5, E = 0.738099333$.

To begin, the approximation domain was first set to $[-7, 7]$, chosen after batch-normalizing the CIFAR-10 dataset and observing that approximately 99.7% of activations fell within $[-3, 3]$, with a small but non-negligible fraction ($\sim 0.029\%$) extending beyond this range. By expanding to $[-7, 7]$, we captured over 99.99% of activations, ensuring the polynomial remained accurate for both common values and rare outliers. Within the selected approximation domain of $[-7, 7]$, region-specific weights were assigned to reflect the relative importance of different subranges in CNN inference. This weighting scheme is motivated by the fact that, after batch normalization, the majority of activation values are concentrated in the central range $[-3, 3]$. Since accuracy in this region has the largest impact on overall model performance, points in $[-3, 3]$ were given the highest priority with a weight of 3. To ensure that less frequent but potentially significant extreme values were still approximated reasonably well, points in the secondary outer regions $[-7, -4]$ and $[4, 7]$ were assigned a moderate weight of 2. All remaining points in the domain were assigned a baseline weight of 1, ensuring global continuity without overemphasizing low-density areas.

The initial polynomial was obtained by solving the weighted least squares problem:

$$\min_{\mathbf{c}} \sum_{i=1}^{N} w(x_i) \cdot \left(p_{\mathbf{c}}(x_i) - f(x_i)\right)^2, \tag{3}$$

where $w(x_i)$ is the region-specific weight for sample point x_i, $f(x_i)$ is the exact activation function value, and $p_{\mathbf{c}}(x_i)$ is the degree-4 polynomial with coefficients $\mathbf{c}$. This formulation ensures that the polynomial fit minimizes the squared error more aggressively in high-priority regions, producing an initial approximation that is both smooth and well-aligned with the activation's critical operating range.

To further improve worst-case performance, the coefficients from the least squares fit were refined using Powell's derivative-free optimization method. Powell's method was applied to solve the weighted minimax problem:

$$\min_{p \in \mathcal{P}_4} \max_{x \in X} w(x) \cdot |f(x) - p(x)|, \tag{4}$$

where $\mathcal{P}_4$ denotes the set of all degree-4 polynomials and X is the set of weighted sampling points over $[-7, 7]$. This step explicitly minimizes the maximum weighted absolute error across all sample points, ensuring that the largest deviation, especially in critical regions, is as small as possible.

Finally, the optimized polynomial was evaluated in an unweighted setting to provide a fair, consistent metric for comparing different activation functions. The unweighted error at each point is defined as:

$$\varepsilon(x) = |p(x) - f(x)|, \quad E_{\max} = \max_{x \in [-7,7]} \varepsilon(x), \tag{5}$$

where $E_{\max}$ represents the maximum absolute error across the entire domain. The $E_{\max}$ values for all candidate activations are reported in Table III, with Softplus achieving the smallest maximum error (0.067), making it the most suitable choice for accurate and efficient encrypted inference.

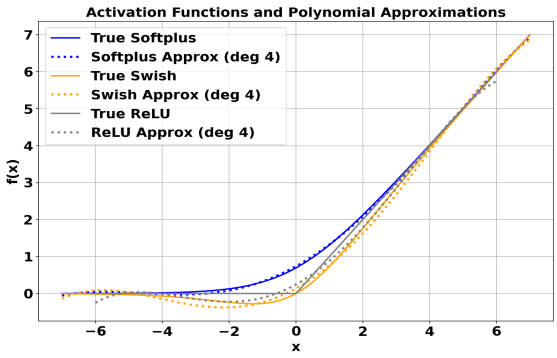

Fig. 2. Swish, ReLU, and Softplus Activation Function Approximations

B. Network Architecture Used in the Training Phase

To evaluate the proposed activation approximation framework, we implemented a convolutional neural network tailored for encrypted inference on the CIFAR-10 dataset. The architecture follows a residual-style design [14] but uses a post-activation block structure, where each convolution is directly followed by batch normalization and the proposed degree 4 polynomial Softplus activation. This ordering can be seen in Fig. 1, and it allows batch normalization to be folded into layers for homomorphic evaluation. The network is composed of four convolutional stages. The first stage applies two 3×3 convolutions with 64 channels, stride 1, and padding 1. The second stage increases the channel count to 128 using two 3×3 convolutions with stride 2 and padding 1, which also downsample the feature maps. The third stage continues the same two-convolution pattern, expanding the channel depth to 256 with stride 2. The fourth stage again applies two 3×3 convolutions, this time with 512 channels and stride 2, further reducing the spatial resolution. After these four stages, a final batch normalization and polynomial activation are applied, followed by a global average pooling layer and a fully connected layer mapping to 10 output units, corresponding to the CIFAR-10 classes. In total, the model contains eight convolution, batch normalization and activation blocks, arranged in pairs across the four stages, with progressively increasing channel sizes of 64, 128, 256, and 512, and strides of 1, 2, 2, and 2 respectively. All nonlinearities use the proposed degree-4 polynomial Softplus approximation with input clamping to reduce approximation error. This design eliminates non-HE-friendly components such as max-pooling and ReLU while maintaining strong representational capacity, ensuring compatibility with homomorphic encryption constraints.

C. Two-Stage Framework for Training and Encrypted Inference

Our methodology employs a two-stage framework designed to enable secure inference under homomorphic encryption (HE) while preserving compatibility with conventional deep learning workflows. In both stages, the network only uses the approximated Softplus activation function, ensuring that the model remains fully HE-compatible from training through deployment. In the first stage, the model described in the 'Network Architecture in the Training Phase' section is trained entirely on plaintext (unencrypted) data. This choice is motivated by two practical reasons. First, it reflects common real-world scenarios in which datasets are already collected and stored in unencrypted form prior to model deployment, making plaintext training both feasible and realistic. Second, performing training without encryption allows us to take advantage of standard GPU acceleration and optimized deep learning libraries, thereby greatly reducing computational costs and training time. Since homomorphic encryption imposes a significant computational overhead, avoiding it during training prevents unnecessary slowdowns and allows for more extensive experimentation and hyperparameter tuning. In the second stage, the trained model is adapted for encrypted inference. This involves replacing plaintext inputs with their CKKS-encrypted counterparts and executing most forward-pass operations in the encrypted domain. By separating training and encrypted inference into distinct phases, we are able to leverage the efficiency and flexibility of raw-data training while confining the computationally intensive HE operations to the inference stage, where privacy preservation is essential. This two-stage design ensures that the model achieves high accuracy and stability during training, while still supporting fully privacy-preserving predictions at deployment, balancing performance, practicality, and security in a single framework.

1) Plaintext Training Phase: In the training phase, the convolutional neural network (CNN) described in the preceding sections was trained on the CIFAR-10 dataset, achieving a final classification accuracy of 94.67%. The dataset was divided into 83.3% for training (50,000 images) and 16.7% for testing (10,000 images). During training, the model learned and stored all parameters required for the encrypted inference stage. These parameters include the weights and biases of the convolutional layers, which extract hierarchical features from the input images, and the parameters of the nine batch normalization layers. Each batch normalization layer stores a scale factor (γ) and shift factor (β), along with running estimates of the mean (μ)

and variance (σ^2) accumulated during training. An additional small constant (ϵ) is used to maintain numerical stability during normalization. For encrypted inference, these batch normalization parameters are not used as separate operations; instead, they are folded into the weights and biases of the fully connected or convolutional layers before encryption, eliminating the need to execute BN steps directly in the encrypted domain and ensuring compatibility with HE constraints. The fully connected layer consist of a layer with 512 units and an output layer with 10 units. The folded weights and biases from the layers is preserved for use during encrypted inference. To reduce the computational complexity under homomorphic encryption, the feature vectors output from the convolutional blocks and batch normalization, just before entering FC1, are pre-computed and stored in plaintext, along with their corresponding ground-truth labels. These pre-computed features are then used as direct inputs to the encrypted inference process, allowing only the fully connected computations to be carried out in the encrypted domain. By retaining the convolutional parameters, batch normalization statistics, folded fully connected weights and biases, pre-computed features, and labels, the system can perform encrypted inference without re-training the model and without recomputing convolutional outputs in ciphertext, thereby reducing multiplicative depth and computational overhead.

2) Encrypted Inference Phase: In this work, we employ the CKKS homomorphic encryption scheme to enable privacy-preserving inference on encrypted data. CKKS is a leveled homomorphic encryption scheme designed for approximate arithmetic on real numbers, making it well-suited for deep learning applications where floating-point operations dominate [15]. In CKKS, vectors are encoded into polynomials over $\mathbb{Z}[X]/(X^N+1)$ and encrypted with a public/secret key pair. Once encrypted, CKKS supports approximate addition and multiplication directly over ciphertexts, enabling the evaluation of linear layers and polynomial activations without decryption. These standard operations are well established in the literature and form the basis for encrypted neural network inference [13], [16].

To eliminate batch normalization (BN) at inference, we fold BN into the preceding layer. If a linear layer produces

$$z = W\mathbf{x} + \mathbf{b}, \tag{6}$$

and the subsequent BN is

$$BN(z) = \gamma \odot \frac{z-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta, \tag{7}$$

then there exist folded parameters W' and $\mathbf{b}'$ such that

$$BN(W\mathbf{x}+\mathbf{b}) \equiv W'\mathbf{x} + \mathbf{b}', \tag{8}$$

with

$$W' = \mathrm{diag}\left(\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}\right) W, \qquad \mathbf{b}' = \frac{\gamma}{\sqrt{\sigma^2+\epsilon}} \odot (\mathbf{b}-\mu) + \beta. \tag{9–10}$$

In our pipeline, FC1 uses the BN-folded parameters $(W^{(1)'}, \mathbf{b}^{(1)'})$. During inference, the TenSEAL CKKS context is configured in standard-precision mode for computational efficiency. Let

$$\mathbf{x}^{(i)} \in \mathbb{R}^d \tag{11}$$

denote the i-th precomputed feature vector. In the encrypted domain, this is encoded and encrypted as

$$c_x^{(i)} = \mathrm{Encrypt}_{pk}(\mathrm{Encode}(\mathbf{x}^{(i)})). \tag{12}$$

The first fully connected layer has weight matrix

$$W^{(1)} \in \mathbb{R}^{h\times d}, \tag{13}$$

and bias vector

$$\mathbf{b}^{(1)} \in \mathbb{R}^h. \tag{14}$$

The j-th neuron output is

$$z_{1,j}^{(i)} = \langle c_x^{(i)}, W_{j,:}^{(1)} \rangle + b_j^{(1)}. \tag{15}$$

Encrypted plaintext dot products are computed using slotwise multiplications and rotations, as supported in TenSEAL [17].

After FC1, the degree-4 polynomial approximation is placed as follows:

$$a_j^{(i)} = -0.00068481\,(z_{1,j}^{(i)})^4 + 0.0887234775\,(z_{1,j}^{(i)})^2 + 0.5\,z_{1,j}^{(i)} + 0.738099333. \tag{19}$$

This polynomial is HE-compatible and avoids costly non-polynomial operations.

The predicted class is obtained as

$$\hat{y}^{(i)} = \arg\max_k \ \ell_k^{(i)}. \tag{23}$$

For a test set of size T, partitioned into batches $\{B_m\}_{m=1}^M$, the accuracy is

$$\mathrm{Acc} = \frac{1}{T}\sum_{m=1}^{M}\sum_{i\in B_m} \mathbf{1}\left\{\arg\max_k \ \ell_k^{(i)} = y^{(i)}\right\}. \tag{24}$$

The encrypted multiplicative depth is dominated by one encrypted plaintext dot product. The polynomial activation is hybrid and thus does not consume encrypted powers. This depth fits comfortably within the noise budget of the standard-precision CKKS setting used here. Running inference on 10,000 samples achieved an accuracy of 94.4% with a total processing time of 2.4 seconds per sample.

IV. Experimental Setup and Results

Experimental Setup: The proposed framework was evaluated using the CIFAR-10 dataset, which contains 60,000 color images across 10 classes with 50,000 training and 10,000 testing. The CNN architecture employed follows the design described in the Methodology, consisting of convolutional blocks with batch normalization folded into the preceding layers, and one fully connected layers, connected via a degree-4 polynomial approximation of the Softplus activation function.
Encrypted inference was implemented using the CKKS scheme through the TenSEAL library, enabling approximate arithmetic over real numbers with SIMD-style batching. To reduce multiplicative depth, only the fully connected layers and polynomial

activation were evaluated in the encrypted domain, while convolutional features were precomputed in plaintext.

All experiments were executed using Intel i9-14900HX CPU with 24 cores and 32 GB RAM. Encryption parameters were selected to balance precision and efficiency, ensuring sufficient noise budget for two encrypted–plaintext dot products per inference.

Results: The proposed framework achieved a classification accuracy of 94.4% on the CIFAR-10 dataset with an inference time of 2.4 seconds for $10{,}000$ encrypted samples. This represents a notable improvement over [5], who reported 89.91‘$\%$ accuracy under a similar degree-4 polynomial approximation, demonstrating the effectiveness of our activation design and architectural modifications.

While higher-degree polynomials may yield closer approximations and potentially higher accuracy, they also increase multiplicative depth and computational cost under homomorphic encryption, requiring either larger noise budgets or costly bootstrapping. Our degree-4 polynomial strikes a practical balance, maintaining high accuracy while ensuring computational feasibility.

In terms of architecture, deeper CNNs could capture richer features but would introduce prohibitive runtime overhead in encrypted inference. Our framework represents a middle ground, complex enough to achieve competitive accuracy on CIFAR-10, yet lightweight enough to keep encrypted inference times within a practical range.

TABLE IV
APPROXIMATION DETAILS OF ACTIVATION FUNCTIONS AND THEIR CORRESPONDING ACCURACIES.

Activation Function	Approximation Degree	RAM and Processing	Accuracy
Softplus	4	Intel® Core™ i9-14900HX (24 cores / 32 threads, 2.20 GHz) with 31.71 GB RAM	94.4%
Softplus [5]	4	Xeon Silver 4114 CPU @ 2.20 GHz with 192 GB RAM	89.91%

V. CONCLUSION

This work presented an optimized framework for privacy-preserving convolutional neural network inference on the CIFAR-10 dataset using homomorphic encryption. By employing a degree-4 polynomial approximation of the Softplus activation function and carefully designing the CNN architecture to balance expressiveness with computational constraints, the system achieved 94.4% accuracy with efficient inference times. The results demonstrate that it is possible to achieve competitive performance without resorting to excessively deep networks or high-degree polynomial approximations, which would otherwise introduce significant computational overhead in the encrypted domain.

REFERENCES

[1] U.S. Congress, “Health insurance portability and accountability act of 1996 (hipaa).” Public Law No. 104-191, 110 Stat. 1936, 1996. Enacted Aug. 21, 1996.

[2] European Parliament and Council of the European Union, “Regulation (eu) 2016/679 (general data protection regulation).” Official Journal of the European Union, L 119, pp. 1–88, 2016. Adopted 27 April 2016; applicable from 25 May 2018.

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

[4] Q. Zhang, C. Xin, and H. Wu, “Gala: Greedy computation for linear algebra in privacy-preserved neural networks,” 2021.

[5] S. Obla, X. Gong, A. Aloufi, P. Hu, and D. Takabi, “Effective activation functions for homomorphic evaluation of deep neural networks,” *IEEE Access*, vol. 8, pp. 153098–153112, 2020.

[6] A. A. Badawi, J. Chao, J. Lin, C. F. Mun, J. J. Sim, 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.” IACR Cryptology ePrint Archive, 2018. Report 2018/1056.

[7] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, “Privacy-preserving classification on deep neural network.” Cryptology ePrint Archive, Paper 2017/035, 2017.

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

[9] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux, “Poseidon: Privacy-preserving federated neural network learning,” 2021.

[10] T. Ishiyama, T. Suzuki, and H. Yamana, “Highly accurate cnn inference using approximate activation functions over homomorphic encryption,” in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 3989–3995, 2020.

[11] Y. Abouelnaga, O. S. Ali, H. Rady, and M. Moustafa, “Cifar-10: Knn-based ensemble of classifiers,” in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 1192–1195, IEEE, 2016.

[12] J. Lee, E. Lee, J.-W. Lee, Y. Kim, Y.-S. Kim, and J.-S. No, “Precise approximation of convolutional neural networks for homomorphically encrypted data,” *IEEE Access*, vol. 11, pp. 62062–62076, 2023.

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

[14] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European Conference on Computer Vision (ECCV)*, pp. 630–645, Springer, 2016.

[15] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Advances in Cryptology – ASIACRYPT 2017* (T. Takagi and T. Peyrin, eds.), vol. 10624 of *Lecture Notes in Computer Science*, pp. 409–437, Springer, Cham, 2017.

[16] S. Bian, Z. Zhao, Z. Zhang, R. Mao, K. Suenaga, Y. Jin, Z. Guan, and J. Liu, “Heir: A unified representation for cross-scheme compilation of fully homomorphic computation.” IACR Cryptology ePrint Archive, 2023. Report 2023/1445.

[17] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, “Tenseal: A library for encrypted tensor operations using homomorphic encryption,” 2021. ICLR 2021 Workshop on Distributed and Private Machine Learning (DPML 2021).