Privacy Preserving Inference for Deep Neural Networks: Optimizing Homomorphic Encryption for Efficient and Secure Classification

Akram, M. A., Khan, F., Tahir, S., Iqbal, A., Shah, S. A. & Baz, A.

DOI 10.1109/access.2024.3357145
ESSN 2169-3536

Publisher: Institute of Electrical and Electronics Engineers

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 December 2023, accepted 17 January 2024, date of publication 22 January 2024, date of current version 1 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3357145

RESEARCH ARTICLE

Privacy Preserving Inference for Deep Neural Networks: Optimizing Homomorphic Encryption for Efficient and Secure Classification

AFTAB AKRAM[1], FAWAD KHAN[1,2], (Senior Member, IEEE), SHAHZAIB TAHIR[1], (Senior Member, IEEE), ASIF IQBAL[3], (Member, IEEE), SYED AZIZ SHAH[2], AND ABDULLAH BAZ[4], (Senior Member, IEEE)

[1]Department of Information Security, College of Signals, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan
[2]Research Centre for Intelligent Healthcare, Coventry University, CV1 5RW Coventry, U.K.
[3]Department of Electrical and Computer Engineering, National University of Singapore, Singapore, Singapore 117583
[4]Department of Computer and Network Engineering, College of Computing, Umm Al-Qura University, Makkah 21955, Saudi Arabia

Corresponding author: Fawad Khan (fawadkhan@mcs.edu.pk)

The authors extend their appreciation to the Deanship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number: IFP22UQU4260426DSR203.

ABSTRACT The application of machine learning in healthcare, financial, social media, and other sensitive sectors not only involves high accuracy but privacy as well. Due to the emergence of the Cloud as a computation and one-to-many access paradigm; training and classification/inference tasks have been outsourced to Cloud. However, its usage is limited due to legal and ethical constraints regarding privacy. In this work, we propose a privacy-preserving neural networks-based classification model based on Homomorphic Encryption (HE) where the user can send an encrypted instance to the cloud and receive an encrypted inference from it to preserve the user's query privacy. In contrast to existing works, we demonstrate the realistic limitations of HE for privacy-preserving machine learning by changing its parameters for enhanced security and accuracy. We showcase scenarios where the choice of HE parameters impedes accurate classification and present an optimized setting for achieving reliable classification. We present several results to demonstrate its effectiveness using MNIST dataset with highly improved inference time for a query as compared to the state of the art.

INDEX TERMS Convolutional neural network, homomorphic encryption, activation function, cloud server, approximation techniques, security and privacy, encrypted computations.

I. INTRODUCTION

Technological advancement and data sharing in the current era of Big Data introduce a new type of concern in everyday human life known as individual privacy. The data being shared by entities and individuals helps learn patterns, draw conclusions, make decisions, and provide ease in most aspects of life. However, this is at the expense of a breach of an individual's privacy and should be catered subject to the GDPR [1] regulations.

Machine Learning (ML), as a technical front-runner that plays a leading role in data analytics, is strongly reliant on the utilization of personal data. Analytical models are used in machine learning to create well-informed predictions on given datasets. Furthermore, a lot of machine learning models require a significant computing resource to analyze enormous volumes of data efficiently.

Cloud on the other hand provides the ease to access data and use on-demand resource sharing from anywhere

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Wang.

in the world. With the expansion of cloud infrastructure, machine learning (ML) models can be trained and deployed on cloud servers. Users may utilize the models to make predictions once they have been deployed, and they don't have to be concerned with the models or the service being maintained. This is what is meant by machine learning as a service (MLaaS), similar to the services provided by Microsoft (Microsoft Azure ML [2]) and Google (Google Prediction API [3]). Both the training and classification phases can be outsourced to the cloud. While performing these phases the ML algorithm, training data, the model, and the feature vector must all be kept secret by one or more of the parties involved in applications that handle sensitive data. One important setting of MLaaS over the cloud involves the cloud having a pre-trained ML model and the client issues a query for inference [4]. This setting is particularly common in healthcare for disease inference, identifying fraudulent transactions by banks, and social media personalized feeds based on clients' previous activities.

Although, the cloud provides numerous advantages in this setting, however, when it comes to security a cloud-based solution invites several problems regarding the individual's privacy. Specifically, for inference or prediction from a pre-trained machine learning model being outsourced to the cloud, the user query privacy is breached. Even if the channel between the user/client and the cloud is secure via traditional means of encryption, the third-party cloud will be able to decrypt user data to properly perform inference operations. However, as the cloud is untrusted, it is a natural practice to encrypt data before outsourcing it to the cloud for performing operations. To cater to this issue, Homomorphic Encryption (HE) is one such alternative to help in cloud settings for performing computations over encrypted data while ensuring client query privacy.

Although the client can issue queries and perform many privacy-preserving inferences from the trained model known as Privacy of Client (PoC), on the other hand, the client should not be able to infer anything beyond the expected output from the trained model stored over the cloud known as the Privacy of Model (PoM). Both PoC and PoM are important metrics for Privacy-Preserving Classification Model (PPCM) in outsourced computation settings. However, attaining these with HE is a difficult task as the underlying functions are non-linear and the biggest challenge is creating the homomorphic version of these functions as inherently the HE schemes [5], [6], [7], [8] supports only addition and multiplication operations.

Some of the HE challenges are summarized as: Firstly, the input data must be encoded as homomorphic plaintexts, and this encoding impacts the efficiency of the resulting circuit. Secondly, HE schemes only support basic arithmetic operations like addition and multiplication and they cannot support high-level functions like rounding, and evaluating non-polynomial functions. Finally, the multiplicative depth of an evaluation function has a direct relation with the HE parameters, so higher depth functions demand bigger HE parameters, which decreases the efficiency of the HE scheme and increases computing time.

Priorly the privacy-preserving classification problem has been addressed in different contexts like validation of 6 non-linear layers [9], compare and contrast several methods and techniques for RELU approximation function [10], [11], acceleration on GPU's [12], and bootstrapping HE with plaintext model training [13]. However, this work examines the behavior of CNN in a privacy-preserving classification setting while considering the HE limitations into consideration. Furthermore, the HE parameters are extensively varied to check the efficiency and accuracy of a given CNN that is being evaluated homomorphically on encrypted data.

A. OUR CONTRIBUTION

This study combines the CNN with HE and uses an open-source Microsoft cryptographic library called SEAL [14], which is built on the BFV [5], [6], [7] and CKKS [8] schemes. Although both encrypted training, as well as encrypted classification, are conceivable, the main purpose of this work is to examine the viability of encrypted classification and the complexities of SEAL. To address the limitations of HE functionality, the non-linear activation functions like ReLU and Sigmoid are approximated to low-degree polynomial approximations. The CNN is trained using the real activation functions on plain data, but the classification phase uses approximated activation functions on HE-encrypted data. Along with the activation layers, the other parts of the network e.g., convolution, pooling, and fully-connected layers are also developed. Initial experiments are carried out with a straightforward three-layer network to ensure that these privacy-preserving layers are correct. Following the success of the preliminary results, a bigger seven-layer network is built to conduct encrypted classification on the handwritten MNIST dataset [15]. The application is designed and tested to ensure accuracy, performance, efficiency, and general applicability. The findings of the experiments highlight the potential importance of HE in modern days cloud-based machine learning information systems, particularly based on CNN. Finally, to explore the security aspect of privacy-preserving classification, the impact of parameter size on efficiency and accuracy is studied by adjusting the HE parameters. We performed comprehensive experiments to demonstrate the efficiency of our model. The following are the key contributions of our study:

- We create a PPCM model that entails training a CNN using plaintext (unencrypted) data while conducting the classification step using homomorphically encrypted data.
- Our approach involves using Chebyshev polynomials to estimate the activation functions (ReLU and Sigmoid), and we then analyze the discrepancies between the

estimated functions in terms of their efficiency and accuracy.
- The security aspect of privacy-preserving classification is being investigated by altering the HE parameters to observe how changes in parameter size impact efficiency and accuracy.
- We present comprehensive results by changing HE parameters, and also illustrate the cases when correct inference cannot be made in the privacy-preserving setting under certain HE parameters.

B. ORGANIZATION

The following is the organization of the paper: Section II presents a review of significant techniques related to PPML. Section III explains the HE scheme and the polynomial approximation techniques. The proposed privacy preserved classification model (PPCM) is demonstrated in Section IV, followed by extensive experimental results in Section V. Lastly, the paper is concluded in Section VI.

II. RELATED WORK

Graepel et al. [16] trained two machine learning models namely Linear Mean and Fisher's Linear Discriminate classifiers using a Somewhat HE scheme. To circumvent HE algorithm's limitations, they introduced division-free algorithms. They did not take into account more complex algorithms but instead concentrated on straightforward classifiers like the linear means classifier. Additionally, they took into account a weak security model and the client can learn the model. Bost et al. [17] proposed a privacy-preserving classification model for three distinct machine learning algorithms named Naive Bayes, Hyperplane Decision, and Decision trees. They combined garbled circuits with three homomorphic encryption schemes named Piallier, BGV, and Quadratic Residuosity schemes and employed Secure Multiparty Computation (SMC) as the basis of their approach, which is effective only for small data sets and only takes into account conventional machine learning techniques.

Xie et al. [18] examined theoretical elements of constructing neural networks in the encrypted domain using polynomial approximation. Gilad-Bachrach et al. [4] extended this work by presenting CryptoNets. It was the first detailed studied CNN classifier for encrypted data. They employed Microsoft SEAL, a leveled homomorphic encryption technique that supported SIMD and scaled mean-pooling layer to solve the division operation limitation, being inaccessible to encrypted values. The sigmoid function was replaced with $f(z) := z^2$ as the activation function in HE schemes. They trained the given model using unencrypted data and utilized it to classify encrypted data. On the MNIST dataset, they got an overall accuracy of 98.95 percent. CryptoNets was able to process 48068 cases per hour. The accuracy of CryptoNets was improved in a study by Chabanne et al. [9] by combining the solution's original concepts with a batch-normalization approach and employing ReLU as the activation function in their scheme.

Jiang et al. [19] proposed a privacy-preserving deep learning model named E2DM (Encrypted Data and Encrypted Model). A matrix was homomorphically encrypted by E2DM before being subjected to arithmetic operations. The primary contribution of this model was the reduction in the complexity required for computing. They employed CNN with a square activation function, two fully connected, and one convolutional layer. To train a simple neural network in privacy preserved environment Aono et al. [20] suggested a technique based on additive homomorphic encryption by pointing out a weakness in Shokri and Shmatikov [21] work, that leaked client data during the training process. The main concept was to allow a server to upgrade the model (learning) by aggregating user gradient values.

Hesamifard et al. [10] proposed a work named CryptoDL, which included a modified version of CNN that operated on encrypted data. The author modified the activation function using low-degree polynomials. This study demonstrated the importance of polynomial approximation of activation functions present in neural networks so that HE operations could be performed on them. They attempted to approximate the ReLU, sigmoid, and tanh types of activation functions. The CNN with polynomial approximation was employed during the training phase. The model created during the training step was then applied to classify encrypted data. This model did not support privacy-preserving deep neural network training on encrypted data. Liu et al. [22] proposed a privacy-preserving technique for CNN training as well as classification purposes. Each activation layer was preceded by a batch-normalization layer, and the activation layer was approximated by using a Taylor series and Gaussian distribution. Additionally, they substituted a convolutional layer with a longer stride for the non-linear pooling layer.

Juvekar et al. [23] introduced a framework named Gazelle, which combined HE with MPC, for privacy-preserving classification purposes. The goal of this study was to retain the model privacy in the server and to make it simpler for the client to perform a classification without exposing his input data to the server. Gazelle effectively blended secret-sharing with HE for privacy-preserving classification since it could switch between HE and GC protocols. The bias, weight, and stride size of the convolutional layer were concealed to protect the neural network model's privacy. Their results demonstrate that, in terms of runtime, Gazelle completely surpasses other well-liked methods like MiniONN [24] and Cryptonets [4].

Sanyal et al. [25] proposed a framework, TAPAS, which employed encrypted data to speed up parallel processing in privacy preserved environment. They tried to overcome the lengthy process of classification in the context of HE. The key contribution was to develop a novel approach to accelerate binary computing in Binary Neural Networks (BNN). Bourse et al. [26] proposed a technique named FHE DiNN (Fast HE Discretized Neural Network) for

privacy-preserving machine learning. They intended to overcome the complexity problem when using a standard HE approach with a neural network. The complexity of the network increases with network depth, which increases the cost of computation. They employed the bootstrapping approach to bring the network complexity to the liner with respect to the depth of the neural network. Badawi et al. [12] developed an effective privacy-preserving solution by combining cutting-edge technology such as Fully Homomorphic Encryption, Convolutional Neural Networks along with Graphics Processing Units (GPUs). The author demonstrated how to use GPUs to increase the performance of running CNNs on encrypted data. They tested two CNNs to classify homomorphically the MNIST and CIFAR-10 datasets and obtained appropriate security (more than 80 bit) and reasonable classification accuracy (99% and 77.55% for MNIST and CIFAR-10, respectively).

Khan et al. [11] presented a technique called Blind Faith which performs the machine learning classification on user-encrypted queries using homomorphic encryption. The primary focus of this work was to elucidate the various methods for approximating activation functions and to discuss the advantages and disadvantages of each of these approximation techniques. They employed the 10-layer convolutional network in their implementation and their model attains the privacy of 98.50% on the MNIST dataset. In a recent study [13], researchers utilized the RNS-CKKS fully homomorphic encryption (FHE) scheme with bootstrapping to implement the standard ResNet-20 model and verified its performance using the CIFAR-10 dataset and plaintext model parameters. Rather than replacing non-arithmetic functions with basic arithmetic functions, advanced approximation methods were utilized to accurately evaluate these non-arithmetic functions, such as ReLU and Softmax. Additionally, the researchers incorporated the bootstrapping technique of the RNS-CKKS scheme for the first time in the proposed model, which enables the evaluation of any deep-learning model on encrypted data.

Zhao et al. [27] proposed a privacy-preserving tree-based inference mechanism by employing Pailliar encryption and demonstrated results for Cancer and Heart datasets. Han et al. [28] employed CKKS for privacy-preserving inference using the Naive Bayes classifier. The system model is considered a central public key shared between the data owner, cloud, and client for encrypting data and performing queries, whereas a system evaluation key is for performing homomorphic operations in the cloud.

Sarkar et al. [29] proposed a fast matrix multiplication algorithm for the high dimensional matrix to optimize performance for privacy-preserving logistic regression. The system model consideredÂ the cloud as an honest entity and PoC is taken into consideration. The results depicted an accuracy of 82.68 for the logistic regression algorithm employing BFV scheme. In another work, Kim and Guyot [30] employed CKKS with bootstrapping and batch convolution for optimization of interference results for CIFAR-10/100 dataset. Table 1 depicts a comparison of existing PPML techniques with the proposed technique.

TABLE 1. Comparative analysis of existing works.

Study	HE Scheme	ML Technique	Dataset	Accuracy (%)	PoC/ PoM
[16]	BFV	Linear Mean & Fisher's Linear Discriminate	Breast Cancer	95.00	PoC
[17]	Paillier, BGV	Hyperplane Decision, Naïve Bayes & Decision Trees	Breast Cancer & ECG	—	PoC
[4]	YASHE	CNN	MNIST	98.95	PoC
[9]	BGV	CNN	MNIST	99.30	PoC
[19]	CKKS	CNN	MNIST	98.10	Both
[20]	Paillier LWE-based	CNN	MNIST & Speech	97.00	Both
[10]	BGV	CNN	MNIST	99.52	PoC
[22]	BGV	CNN	MNIST	98.97	PoC
[23]	HE + MPC	CNN	MNIST & CIFAR-10	—	Both
[25]	TFHE	BNN	Cancer & MNIST	98.60	Both
[26]	TFHE	DiNN	MNIST	96.35	Both
[12]	BFV	CNN	MNIST & CIFAR-10	99	PoC
[11]	BFV	CNN	MNIST	98.5	PoC
[13]	RNS-CKKS	CNN ResNet-20	CIFAR-10	98.43	PoC
[27]	Paillier	XGBoost	Heart	81.6	Both
[28]	CKKS	Naive Bayes	Breast Cancer	97.8	Both
[29]	BFV	LR	TCGA	82.68	PoC
[30]	CKKS	CCN	CIFAR-10	94	PoC
This Work	BFV	CNN	MNIST	98.55	PoC

III. PRELIMINARIES

A. HOMOMORPHIC ENCRYPTION

The FV [7] technique is the homomorphic encryption scheme employed by the Microsoft SEAL [14] library and it is based on the algebraic ring structure. Basically, algebraic rings are mathematical sets of elements inside a modulus that simultaneously enable the binary operations of addition and multiplication. To make the FV scheme work, initial plaintext numbers must be obtained in the ring structure R_t. The ring R_t is defined as $R_t = Z_t[x]/x^{n+1}$, which includes only those integer numbers from the set of integers Z for which there exists a polynomial having degree less than n with coefficients reduced modulo t [31]. The ring structure permits polynomials with coefficients modulo t and a degree less than n. The t and $x^n + 1$ are referred to as the plaintext and polynomial moduli, respectively. The encryption process begins with the specification of both of these moduli as encryption parameters. Since each of the original numbers must be a member of the ring structure R_t in order to be encryptable under this scheme, hence any number whether it be an integer or a rational number must be encoded into a plaintext polynomial in R_t before it can be encrypted under the scheme, according to the ring R_t. After the appropriate integers have been encoded into R_t they are encrypted into a ciphertext array of at least two polynomials in the ring structure R_q where q is the coefficient modulus and is specified as an encryption parameter before the encryption occurs.

We briefly overview the algorithms that are part of the FV scheme. A $a \leftarrow R_2$ denotes that a is sampled uniformly from the finite set R_2. The scheme's main algorithms are as follows:

- Setup $(pk, sk, evk \leftarrow \lambda)$: The algorithm takes as input the security parameter λ to generate the public, secret, and evaluation keys respectively as pk, sk, evk.
- Encryption $(ct \leftarrow pk, m)$: The algorithm takes the public key $pk = (p_0, p_1)$, a message $m \in R_t$ and sample $e_1, e_2 \leftarrow X$ and $u \leftarrow R_2$ as random inputs and generates the ciphertext ct as $([\Delta m + p_0 u + e_1]_q, [p_1 u + e_2]_q)$.
- Decryption $(m \leftarrow sk, ct)$: The algorithm takes secret key $s = sk$, ciphertext $c_0 = ct[0]$ and $c_1 = ct[1]$ as inputs to compute $m' = [\lfloor \frac{t}{q}[c_0 + c_1 s]_q \rceil]_t$ to get the decryption of m as m'.
- Evaluation $(c' \leftarrow pk, ct_1, ct_2)$: This algorithm takes two ciphertexts (ct_1, ct_2) and public key as input to output a resulting ciphertext c', which is either the addition or multiplication of given ciphertexts (ct_1, ct_2) as specified to the algorithm.

The FV scheme allows for both addition and multiplication operations to be performed in the encrypted domain. It also supports Single Instruction Multiple Data (SIMD), which allows for HE operations to be performed on batches of ciphertexts [32] to boost the performance. There are three main parameters associated with this scheme that has a direct impact on the security level as well as the performance of that scheme: polynomial modulus (n), coefficient modulus (q), and plaintext modulus (t). The polynomial modulus affects the security level and performance of the scheme, while the coefficient modulus tells about the Noise Margin, also known as Noise Budget (NB) available for the encrypted computations. On the other hand, the plaintext modulus provides the NB present in the freshly encrypted ciphertext and it also provides NB consumption during the encrypted multiplications. The NB is consumed by homomorphic operations and is based on the chosen encryption parameters. The consumption of NB is higher in sequential multiplication and can be reduced by choosing reasonable encryption parameters. The decryption of ciphertext is impossible if the NB falls to zero, hence it's important to set parameters that are large enough to prevent this but not too big that they lose their effectiveness and functionality.

B. POLYNOMIAL APPROXIMATION

The HE scheme discussed above allows to perform only addition and multiplication operations, i.e., it can only handle linear mathematical functions that contain addition and multiplication terms and fails to work with nonlinear functions. Hence, the nonlinear functions must be converted to functions that only involve addition and multiplication terms to make them compatible with the HE scheme. For this purpose, different polynomial approximation techniques are used and given as follows:

- Numerical Approximation Method
- Taylor Series Method
- Chebyshev Approximation Method

We employ each of these techniques individually to analyze the polynomial approximation of non-linear activation functions present in the neural network. Experiments show that Numerical Approximation only works best for higher degree polynomial approximation which leads to be ineffective when dealing with encrypted domains using the HE scheme. For a lower-degree polynomial approximation, the accuracy of the activation function decreases. On the other hand, Taylor Series [33], [34] was also found to be ineffective due to two key problems. The first problem is that, despite being lower than the above method, the high degree of polynomial approximation is still too high to be used with HE schemes. Secondly, the approximation interval is the most crucial problem. The fundamental goal of this series is to make an approximation of given functions in a point nearby space. The approximation error is significantly larger for the points outside of the input interval than for those within it. For instance, this approach is unable to cover the [0,255] range of integer values for pixels in the MNIST dataset.

The use of Chebyshev polynomials [34] is not as widespread as earlier techniques. However, these are more appropriate because we can estimate a function throughout an interval rather than just a tiny region around a point. We were able to cover integers since HE schemes are over integers with message space Z. The Chybeshev polynomial is given as:

$$T_{(n+1)}(x) = 2xT_n(x) - T_{(n-1)}(x) \tag{1}$$

The minimax approximation is another name for the Chebyshev approximation. By increasing accuracy and reducing overall computing cost, the minimax polynomial technique is employed for function approximation. As opposed to Taylor's polynomial approximation, which minimizes error at the point of expansion, the minimax technique reduces error across a specific input segment.

IV. PRIVACY PRESERVING CLASSIFICATION MODEL (PPCM)

A. SYSTEM AND THREAT MODEL

To maintain the privacy of the major components of the privacy-preserving classification model (PPCM) as a service framework, we take into account the system model shown in Fig 1, in which the cloud server has an unencrypted (plain) CNN model that has been trained to classify the client's unseen instances. The classification process in this case only works with user-provided encrypted data, and the system model is similar to [4], [10]. The client gives encrypted instances to the cloud server which classifies these instances and returns back the encrypted results to the client. The intent of this system model is to protect the privacy of feature values of the client's input as well as the predicted values of unseen instances from the server known as PoC. Both the client/user and cloud are considered honest but curious, i.e., both will follow the protocol but will try to infer as much information as they can from the interaction.

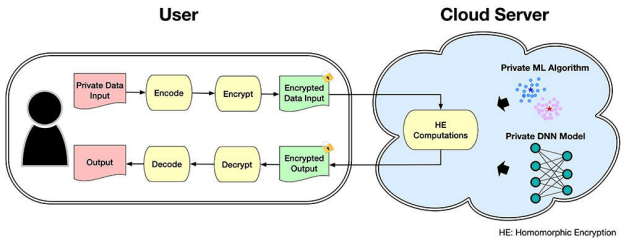


FIGURE 1. PPCM system model.

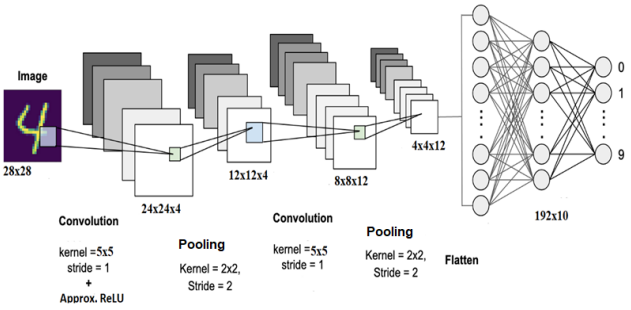


FIGURE 2. CNN network.

B. CNN NETWORK

The CNN network which is used to train and categorize the MNIST dataset is shown in Fig 2. A summary of this network is provided below:

- **1st Convolution-Layer**: It takes an image of dimension 28×28×1 as an input. The layer contains 4 kernels of dimension 5×5, with a stride of (1,1).
- **Activation-Layer**: It performs the ReLU function at every input node and is present after every convolution layer.
- **1st Pooling-Layer**: It takes an input of dimension 24×24×4 and have stride of size (2,2). Its output is 12×12×4.
- **2nd Convolution-Layer**: It has input of dimension 12×12×4. This layer consists of 12 kernels of dimension 5×5 and a stride of (1,1). The outcome of this layer is 8×8×12.
- **2nd Pooling-Layer**: It takes an input of dimension 8×8×12 and have stride of size (2,2). Its output is 4×4×12.
- **Flatten-Layer**: The input of this layer is 4×4×12 and returns the output of 192.
- **Fully Connected-Layer**: It combines incoming 192 nodes to the 10 output nodes.

C. PPCM LAYER DESIGN

The primary objective of this research work is to solely perform classification on encrypted data. Hence, the CNN layers are designed while considering the feed-forward network only, and the back-propagation phase is omitted. The only operations supported by the HE scheme are addition and multiplications, so CNN layers are designed while keeping these limitations in mind. Before creating a privacy-preserving CNN model, all the layers of CNN are researched, implemented, and then tested both in plaintext as well as ciphertext space. The plain layers of CNN are used as a reference only for comparison to verify the accuracy of the encrypted classification. Below, we discuss the modifications incorporated into CCN layers for privacy-preserving inference settings.

1) ACTIVATION FUNCTIONS DESIGN

In contrast to the layers in Fig 1, there is another layer that contains a non-linear function commonly known as the activation layer. This layer usually comes after each convolutional layer. Each neuron in the preceding layer is activated using a nonlinear activation function. This layer adds a non-linear component to CNN, allowing them to solve more complicated classification problems. The following equations represent the Sigmoid and ReLU functions.

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2}$$

$$\text{ReLU}(z) = \begin{cases} Z, & z > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Since it is obvious that these equations involve nonlinear functions, we must change them to make them compatible with HE schemes. So, to combat this challenge, different polynomial approximation techniques are explored as discussed in III-B. By doing analysis it is found that the Chebyshev approximation technique is best for approximating the activation functions. We employ the Chebyshev polynomial approximate method to approximate both the ReLU as well as Sigmoid activation functions in our proposed model, in which the inputs are HE-encrypted images.

Table 2 and Fig 3 depict the polynomial approximation of the ReLU activation function with degrees 5 and 7. Since the degree and interval choices have an impact on the model's performance, it is necessary to select appropriate parameters. To achieve this, we ran several experiments with various intervals and degrees. Table 2 illustrates that by using higher degree polynomials in shorter intervals, the activation functions can be more accurately approximated. Specifically, the polynomial with a degree of 7 and interval of [10,10] provides a more precise approximation of the ReLU function compared to other polynomials. This is also true for the Sigmoid activation function, as shown in Table 3. However, using higher-order polynomials incurs significant computational overhead and short intervals limit the applicability of approximated functions. Similarly, Table 3 and Fig 4 demonstrate the approximation of the Sigmoid function.

A similar study by Khan et al. [11] utilized the same method for polynomial approximation of activation function and they employed the degree five polynomial in their implementation, however, a higher degree polynomial approximation of activation function leads to being

TABLE 2. Approx. of ReLU function on two intervals using degree (ψ) 5 & 7.

ψ	Intervals	Polynomial Approximation	ReLU(z)
5	[-10,10]	$(2.368475785867 \times 10^{-19}) \times x^5 - (0.00025262492130867 4) \times x^4 \ldots$	Fig 3a
7	[-10,10]	$(-8.88178419700125 \times 10^{-21}) \times x^7 + (3.66197231323541 \times 10^{-6}) \times x^6 \ldots$	Fig 3b
5	[-100,100]	$(2.27373675443232 \times 10^{-23}) \times x^5 - (2.52624921308674 \times 10^{-7}) \times x^4 \ldots$	Fig 3c
7	[-100,100]	$(-6.82121026329696 \times 10^{-27}) \times x^7 + (3.6619723132354 \times 10^{-11}) \times x^6 \ldots$	Fig 3d

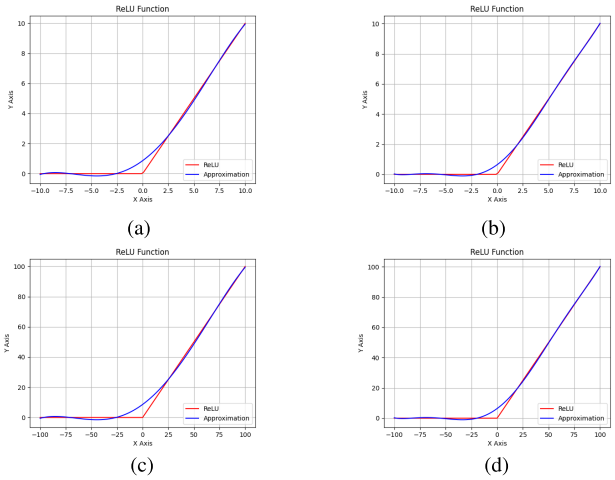


FIGURE 3. Approximation of Relu functions.

TABLE 3. Approx. of sigmoid function on two intervals using degree (ψ) 5 & 7.

ψ	Interval	Polynomial Approximation	$\sigma(z)$
5	[-10,10]	$(2.04674243304457 \times 10^{-5}) \times x^5 + (2.46800554530117 \times 10^{-20}) \times x^4 \ldots$	Fig 4a
7	[-10,10]	$(-4.3491363562486 \times 10^{-7}) \times x^7 - (2.72617215260618 \times 10^{-21}) \times x^6 \ldots$	Fig 4b
5	[-100,100]	$(2.76073648103432 \times 10^{-10}) \times x^5 - (5.79639277612257 \times 10^{-24}) \times x^4 \ldots$	Fig 4c
7	[-100,100]	$(-8.15672915997047 \times 10^{-14}) \times x^7 - (2.69310298657131 \times 10^{-27}) \times x^6 \ldots$	Fig 4d

computationally ineffective while performing calculations in the encrypted domain. So, we approximated algorithms of both the ReLU and Sigmoid activation function with two degrees of approximation in our design of PPCM as described below in Algorithm 1 and 2 respectively.

2) CONVOLUTIONAL LAYER DESIGN

The convolutional layer contains a sliding filter that is applied to the input image. When a filter is applied to an image it extracts a certain feature from it. Therefore, many filters may be applied to the same layer to extract various features of an image. The three-dimensional sliding filter is made up of the number of weights that are learned throughout the training. Each filter is a $n \times n$ square (e.g., $n = 3$ or 5) with a stride. The stride is a set of two integers, e.g., a stride of (2, 2) means that at each step a filter is moving two units to the right or down. This layer calculates the dot-product between filter weights and associated values in the pixel's neighbor by convolving the pixels in the image. As this layer is linear and just requires addition and multiplication operations for convolution, hence

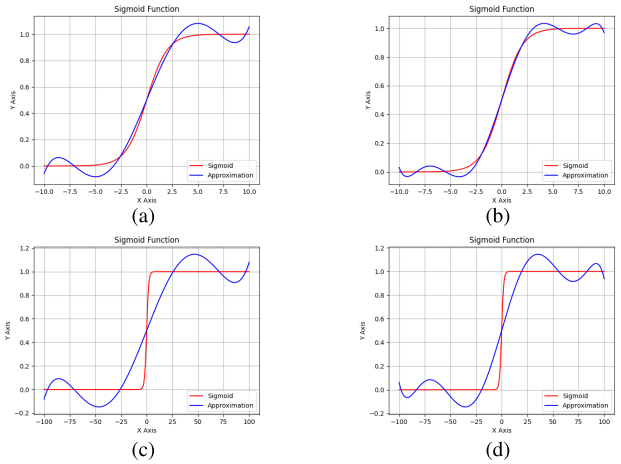


FIGURE 4. Approximation of sigmoid functions.

Algorithm 1 ReLU Function

Input : In, In_{input}
Output: Out
$Scale \leftarrow 10000$;
$a_0 \leftarrow 45000$;
$a_1 \leftarrow 5000$;
$a_2 \leftarrow 55$;
$In_{size} \leftarrow In_{input}$;
for $j \leftarrow 0, 1$ *to* In_{size} **do**
  $Out[j] \leftarrow In[j] * In[j] * a_2 + In[j] * a_1 + a_0$
end

Algorithm 2 Sigmoid Function

Input : In, In_{input}
Output: Out
$Scale \leftarrow 10000$;
$b_0 \leftarrow 5000$;
$b_1 \leftarrow 700$;
$b_2 \leftarrow -300$;
$In_{size} \leftarrow In_{input}$;
for $j \leftarrow 0, 1, \ldots, In_{size}$ **do**
  $Out[j] \leftarrow In[j] * In[j] * b_2 + In[j] * b_1 + b_0$
end

one can simply transform this layer into the encrypted domain easily by employing the HE scheme without any changes.

3) POOLING LAYER DESIGN

The pooling layer is used to decrease the size of the data by sub-sampling it. This layer usually comes after the activation layer and is non-linear. The two most common non-linear pooling layers are the Average and Max pooling layers. The maximum value inside the subsection is the outcome of the Max pooling layer while an average of all values inside the subsection is the outcome of the Average pooling layer. The Max and Average Pool layers have to be changed because HE does not offer any division and comparison operations. To overcome this limitation, we replaced these layers with a sum-pooling layer [35] as shown in Algorithm 3. The output of the sum-pooling layer is simply the summation of values within the sliding window and can be realized by a homomorphic addition operation

Algorithm 3 Sum Pooling Layer

```
Input  : In, In_height, In_width, Depth, Poolx, Pooly
Output: Out
Count ← 0;
Out_height ← In_height/Pooly;
Out_width ← In_width/Poolx;
for a ← 0, 1, ..., Depth do
    for b ← 0, 1, ..., Out_height do
        for c ← 0, 1, ..., Out_width do
            for j ← 0, 1, ..., Pooly do
                for i ← 0, 1, ..., Poolx do
                    Temp ←
                     In[b × Pooly + j][c × Poolx + i][a];
                    if Count = 0 then
                        out[b][c][a] ← Temp;
                        Count ← Count + 1;
                    end
                    else
                        Out[b][c][a] ←
                         Out[b][c][a] + Temp;
                        Count ← Count + 1;
                    end
                    if Count = Pooly × Poolx then
                        Count ← 0
                    end
                end
            end
        end
    end
end
```

4) FULLY CONNECTED LAYER DESIGN

It is usually the last layer of CNN. The number of classes in the dataset is the output of this layer. Every neuron present in this layer is linked to all other neurons of the preceding layer. The total sum of weights in this layer is the product of the total number of neurons in the preceding and current layers. This layer calculates the input vector's dot product with weight vector, then adds bias vector to outcome elementwise as $z = w^T \cdot x + b$. As this step just requires the addition and multiplication operations as seen from the equation, hence it can be transformed into the encrypted homomorphic domain easily for privacy preservation.

V. PERFORMANCE AND EVALUATION

This section contains the experimental results which were obtained through research and analysis.

A. SETUP

All the experiments were carried out in Python-3 on Windows 10 pro-64-bit (Intel Core i5, 1.60 GHz, 8GB). During the training period, we employed the PyTorch framework in VS code setup to train the given CNN model. Finally, for the inference phase, we augmented the Python version of SEAL, called Pyfhel [36] for privacy-preserving model classification.

B. DATASET

The MNIST [15] data set is employed to train as well as test the PPCM. This dataset is chosen specifically because it is widely used in the field of deep learning and enables a comparison of accuracy with prior research works. There are 60,000 total images in this dataset, from which 50,000 images are chosen for training while the remaining 10,000 images are selected for testing. The MNIST dataset contains images of 28×28 pixel arrays, every pixel consists of a positive integer ranging from 0-255.

C. MODEL TRAINING

The training is carried out in batches of 128 for a total of 1000 epochs. The Adaptive Moment Estimation, often known as Adam, is the optimization technique utilized during training. Adam is chosen because it requires less memory and performs well with minimal hyperparameter adjustment. We conduct our experiments using the MNIST dataset and the CNN model that is described in Fig 2 to assess the performance of various approximation methods. For comparison, we first train the given model using original activation functions and find the accuracy against each activation function as shown in Table 4. Then we use the approximation polynomial of degree two for the given activation functions in the given model and calculate the accuracies given in Table 4. We conclude that ReLU provides more accuracy for both the original and approximated models and hence is employed in PPCM for the presentation of further results.

D. INFERENCE ON ENCRYPTED DATA

The model uses an encrypted PNG image of a handwritten digit from 0 to 9 as input, and the weights are determined during training. The encrypted image is then classified, and

TABLE 4. CNN accuracies based on original and approximated activation functions.

Activation Function	Original Model	Approximated Model
Sigmoid	98.82 %	98.25%
ReLU	99.15%	98.55%

the final layer output is decrypted. The output vector has ten values, and each of them corresponds to a digit from 0 to 9. The classifier's prediction is whatever number from 0 to 9 is connected with the highest value discovered in the output. Furthermore, images are not classified in batches since working with encrypted data requires a considerable amount of processing power and memory. Instead, the privacy-preserving classier processes each image separately.

E. SECURITY AND PERFORMANCE METRICS

Privacy, accuracy, and the incurred computational time are the three important metrics to examine when evaluating the practicality of any privacy-preserving machine learning technique. Although privacy is ensured by HE, however, this is at the expense of an increase in computational time and a reduction in accuracy. We put our privacy-preserving CNN model to the test under various circumstances to see how different factors affect accuracy and time in an effort to better understand and present the capabilities and limitations of HE. The running time is determined by counting the seconds it took to perform operations at each layer. To determine the accuracy, the model is typically run over the complete test dataset, however, due to resource constraints, a very simple test has to be constructed. The privacy-preserving CNN is applied to a single random picture from the testing dataset rather than testing all images at once.

1) EACH LAYER EXECUTION TIME

It was discovered during the initial testing phases that our privacy-preserved CNN model takes 215.08 seconds to classify an encrypted image. We evaluated the time taken by each layer to determine where it might be spending most of its time during the classification phase. The HE security parameter is set to 128 bits for the evaluation of time for each layer of our CNN model and is presented in Table 5. As seen from Table 5, the convolution layer requires the most computation time to execute. The activation layer is the 2nd most expensive layer in terms of time, then comes the fully connected layer, and lastly the pooling layer.

2) SECURITY PARAMETER λ VARIATION

From the perspective of model security, HE has a few factors that are essential to consider while performing the CNN classification. One of these parameters is λ, which is called the security parameter. We use the default value of $\lambda = 128$ for experiments. The security parameters are varied in this section to evaluate the execution time and classification accuracy as well as the overall security. The timing values are calculated by using the BFV scheme for polynomial modulus degree $n = 8192$, for each of three security parameters of 128, 192, and 256 as recommended in [37]. For SEAL, setting $\lambda = 128$ is comparable to AES 128-bit security, setting $\lambda = 192$ is equivalent to AES 192-bit security, and setting $\lambda = 256$ is similar to AES 256-bit security. Therefore, in addition to the default $\lambda = 128$, these are the two other security settings assessed. Fig 5 depicts the time each CNN layer takes to run based on the variation in security parameters. According to the stats in Fig 5, the time required to evaluate each layer increases as the security parameter increases. When the security parameter is increased from 128 bits to 256 bits, the evaluated time for the layers (Conv1/Conv2/FC1) increases approximately by 1.8 times.

TABLE 5. Running time cost of CNN layers.

Layers	Description	Time(s)
1st Convolutional Layer	Input: 28x28x1 Output: 24x24x4	79
Activation Layer (ReLU)	2nd Degree Polynomial	25
1st Pooling Layer	Avg. Pooling	0.01
2nd Convolutional Layer	Input: 28x28x1 Output: 24x24x4	106
2nd Pooling Layer	Avg. Pooling	0.01
Flatten Layer	Output: 192	0.06
Fully Connected Layer	Output: 10	2

TABLE 6. Encryption/Decryption time based on security parameters.

Security Param.	Polynomial Modulus	Encryption Time	Decryption Time	Prediction
128	8192	3.28	0.02	YES
192	8192	3.45	0.025	YES
256	8192	3.98	—	NO

Based on the sizes of the security parameter, Table 6 indicates the time taken by the model to encrypt and decrypt the given image. It also determines whether or not the image is properly classified. According to the timing data in Table 6 security parameter change also has an influence on the encryption and decryption time of the image. The time required for the encryption of an image increases with the increase in the security parameter. However, in comparison to classification, this time difference is trivial because it is only a few seconds.

Accuracy is also impacted by changes in security parameters. The 256-bit security parameter is improperly classifying the encrypted image since there aren't enough levels available to accommodate it. To resolve this issue, the polynomial modulus degree size is increased from 8192 to 16384, while keeping the same security level of 256.

3) OTHER PARAMETERS (*Q* & *N*) VARIATIONS

In this section, both the values of coefficient modulus q and polynomial modulus n are varied to observe the change in threshold values and overall time. All timing values are measured by using FV [7] scheme for different values of n (1024, 2048, 4096, 8192, 16384) with standard security parameters of 128-bit. Fig 6 illustrates the total execution

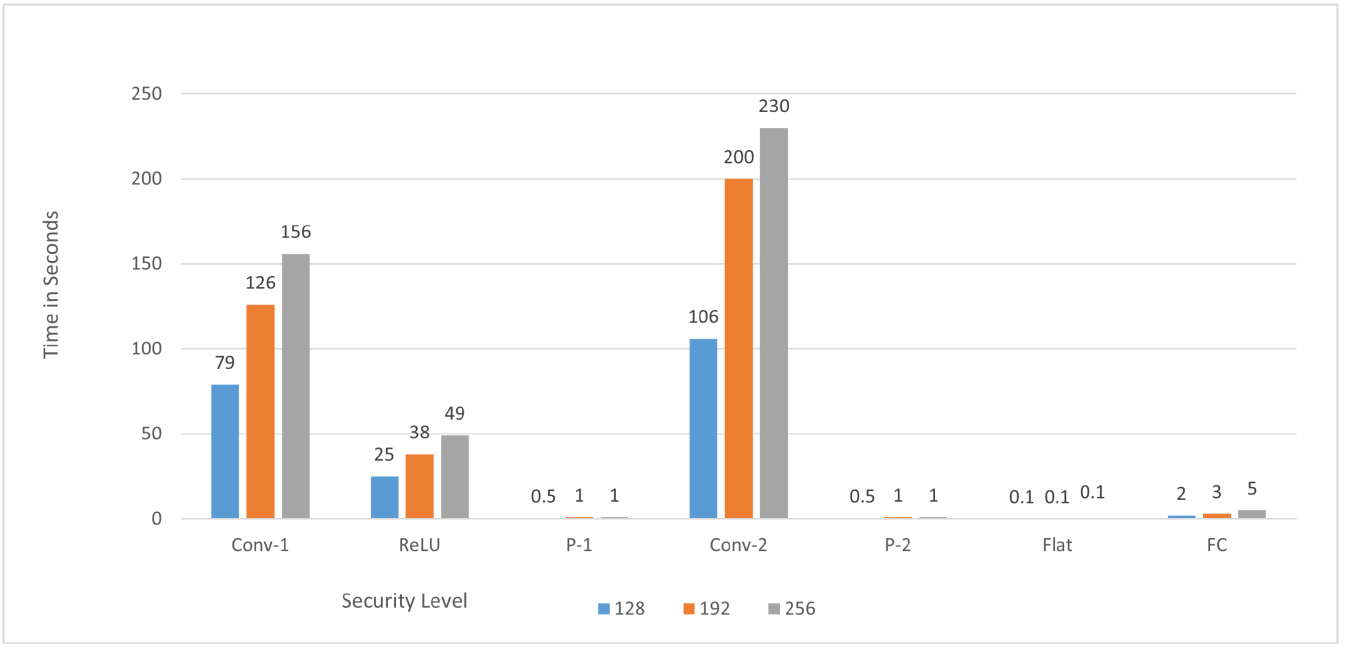


FIGURE 5. Execution Time of different layers based on security parameters variation.

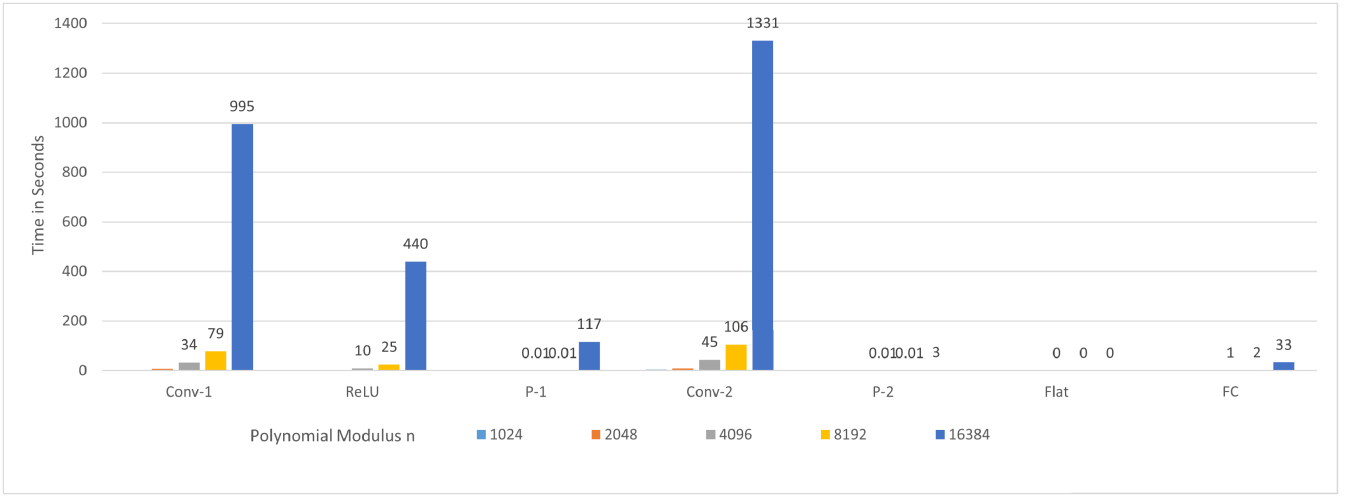


FIGURE 6. Execution time of different layers based on polynomial degree variation.

time of each layer in our network based on variation in polynomial modulus degree n and the corresponding number of levels in coefficient modulus q. As the value of the polynomial modulus degree increases, the time of execution of each layer increases too. When the value of polynomial degree n is changed from 1024 to 16384, there is a 2x-6x increase in calculation time for the layers (Conv/ReLU/FC).

Table 7 shows the time to encrypt as well as decrypt the given image and determine if the given image is classified correctly depending on the value of polynomial modulus degree n along with the corresponding levels in coefficient modulus q. The timing cost in Table 7 clearly shows that polynomial modulus degree n and coefficient modulus q do affect the encryption and decryption time of an image. This time increases as the values of polynomial modulus degree n and coefficient modulus q increase.

Accuracy is also affected by varying the size of polynomial modulus degree n and coefficient modulus q. A certain minimum number of levels corresponding to the polynomial modulus degree n is undoubtedly required to categorize the given encrypted image and control the NB effectively. There is no simple formula for calculating the required number of levels in coefficient modulus q. However, in general, the number of levels must be equivalent to the count of multiplications in the evaluation function. The network put to the test in this experiment includes a degree of two

TABLE 7. Encryption/Decryption time based on polynomial and Ciphertext modulus.

Polynomial Modulus Degree	Ciphertext Modulus Levels	Encryption Time (s)	Decryption Time (s)	Prediction
1024	128	0.34	—	NO
2048	128	0.66	0.01	NO
4096	128	1.53	0.01	YES
8192	128	3.72	0.03	YES
16384	128	14.83	0.34	YES

TABLE 8. Timing cost of each layer for optimized configuration.

Conv-1	ReLU	P-1	Conv-2	P-2	Flat	FC
34	10	0.01	45	0.005	0.005	1

polynomial computations and three dot products due to which early tests were conducted with different levels set. The minimal number of levels was found by guessing and checking when the results displayed an error message.

4) OPTIMIZED CONFIGURATION

By understanding all the above-mentioned results and limitations of HE, the final test for the fast configuration is carried out. Best configuration means selecting such parameters that yield both accurate predictions and the fastest timing results. In this section, all the timing values are calculated by taking the polynomial modulus degree value of 4096 and the security parameter set to 128-bit. Table 8 represents the all-time best value of each layer execution time based on the best parameter selection, while still maintaining the correct prediction. We remark that if the model is encrypted, then the inference cost is higher because both the query and model are in ciphertext domain.

VI. CONCLUSION

In this paper, we looked into ways to outsource computing securely while employing homomorphic encryption on encrypted data. We suggested cryptographic protocols for the widely used algorithms to act as building blocks to allow a wide range of secure data analytics and machine learning applications on the cloud. The HE limitations for privacy-preserving machine learning were investigated, specifically for employing CNN for classification in an outsourced setting. To keep these limitations in mind, we modified the different layers of CNN to make them compatible with HE-supported operations. In particular, we approximated the non-linear activation functions like Sigmoid and ReLU into functions that only include additions and multiplications terms. We approximated these functions on different degrees and scales to examine the impact of these variations on the accuracy of a proposed classification model. In the end, we calculated the time of each layer and the overall time as well as the accuracy of the proposed model by varying the HE parameters. Overall, this work served as an effective demonstration of the concept for the classification of encrypted images.

CONFLICT OF INTEREST

The authors declare no conflicts of interest.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number: IFP22UQU4260426DSR203.

REFERENCES

[1] M. Paterson and M. McDonagh, "Data protection in an era of big data: The challenges posed by big personal data," *Monash Univ. Law Rev.*, vol. 44, no. 1, pp. 1–31, 2018.
[2] Microsoft. *Azure Machine Learning*. Accessed: Mar. 20, 2023. [Online]. Available: https://azure.microsoft.com/enau/products/machine-learning
[3] Google. *Google Prediction API*. Accessed: Apr. 1, 2023. [Online]. Available: https://cloud.google.com/vertexai/docs/predictions/get-predictions
[4] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.
[5] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Proc. Annu. Cryptol. Conf.* Cham, Switzerland: Springer, 2012, pp. 868–886.
[6] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 1–36, Jul. 2014.
[7] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," Cryptol. ePrint Arch., Paper 2012/144, 2012. [Online]. Available: https://eprint.iacr.org/2012/144
[8] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Cham, Switzerland: Springer, 2017, pp. 409–437.
[9] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," Cryptol. ePrint Arch., Paper 2017/035, 2017. [Online]. Available: https://eprint.iacr.org/2017/035
[10] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep neural networks over encrypted data," 2017, *arXiv:1711.05189*.
[11] T. Khan, A. Bakas, and A. Michalas, "Blind faith: Privacy-preserving machine learning using function approximation," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Sep. 2021, pp. 1–7.
[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. Topics Comput.*, vol. 9, no. 3, pp. 1330–1343, Jul. 2021.
[13] 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.
[14] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library-SEAL v2. 1," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, Apr. 2017, pp. 3–18.
[15] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[16] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Cham, Switzerland: Springer, 2012, pp. 1–21.
[17] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, p. 4.
[18] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-Nets: Neural networks over encrypted data," 2014, *arXiv:1412.6181*.
[19] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1209–1222.
[20] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.
[21] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1310–1321.

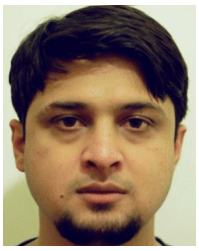
[22] W. Liu, F. Pan, X. A. Wang, Y. Cao, and D. Tang, "Privacy-preserving all convolutional net based on homomorphic encryption," in *Proc. Int. Conf. Netw.-Based Inf. Syst.* Cham, Switzerland: Springer, 2018, pp. 752–762.
[23] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1651–1669.
[24] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via MiniONN transformations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 619–631.
[25] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade, "TAPAS: Tricks to accelerate (encrypted) prediction as a service," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 4490–4499.
[26] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Proc. Annu. Int. Cryptol. Conf. (CRYPTO)*. Cham, Switzerland: Springer, 2018, pp. 483–512.
[27] J. Zhao, H. Zhu, F. Wang, R. Lu, and H. Li, "Efficient and privacy-preserving tree-based inference via additive homomorphic encryption," *Inf. Sci.*, vol. 650, Jan. 2023, Art. no. 119480.
[28] B. Han, Y. Kim, J. Choi, H. Shin, and Y. Lee, "Fully homomorphic privacy-preserving naive Bayes machine learning and classification," in *Proc. 11th Workshop Encrypted Comput. Appl. Homomorphic Cryptogr.*, Nov. 2023, pp. 91–102.
[29] E. Sarkar, E. Chielle, G. Gursoy, L. Chen, M. Gerstein, and M. Maniatakos, "Privacy-preserving cancer type prediction with homomorphic encryption," *Sci. Rep.*, vol. 13, no. 1, p. 1661, Jan. 2023.
[30] D. Kim and C. Guyot, "Optimized privacy-preserving CNN inference with fully homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2175–2187, 2023.
[31] R. Thangadurai, *On the Coefficients of Cyclotomic Polynomials*. Pune, India: Cyclotomic Fields Rel. Topics, 2000, pp. 311–322.
[32] S. Halevi and V. Shoup, "Algorithms in HElib," in *Advances in Cryptology—CRYPTO 2014*. Cham, Switzerland: Springer, 2014, pp. 554–571.
[33] W. Han and K. E. Atkinson, *Theoretical Numerical Analysis: A Functional Analysis Framework*. Cham, Switzerland: Springer, 2009.
[34] C. F. Dunkl and Y. Xu, *Orthogonal Polynomials of Several Variables*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[35] P. Rama, *Exploring the Effectiveness of Privacy Preserving Classification in Convolutional Neural Networks*. Rochester, NY, USA: Rochester Inst. Technol., 2019.
[36] A. Ibarrondo and A. Viand, "Pyfhel: Python for homomorphic encryption libraries," in *Proc. 9th Workshop Encrypted Comput. Appl. Homomorphic Cryptogr.*, Nov. 2021, pp. 11–16.
[37] M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, J. Hoffstein, K. Lauter, S. Lokam, D. Moody, and T. Morrison, "Security of homomorphic encryption," HomomorphicEncryption, Redmond, WA, USA, Tech. Rep., Jan. 2018. [Online]. Available: https://www.microsoft.com/en-us/research/wpcontent/uploads/2018/01/security_homomorphic_encryption_white_paper.pdf

AFTAB AKRAM received the master's degree in information security from the National University of Sciences and Technology (NUST), Islamabad, Pakistan, in 2023, under the supervision of Fawad Khan. His research interests include privacy-preserving deep learning, homomorphic encryption, multiparty computation, and federated learning.

FAWAD KHAN (Senior Member, IEEE) received the Ph.D. degree in information security from the School of Cyber Engineering, Xidian University, Xi'an, China, in 2018. Currently, he is a Research Fellow with the Research Centre for Intelligent Healthcare, Coventry University, Coventry, U.K. He was an Assistant Professor with the Department of Information Security, National University of Sciences and Technology (NUST), Islamabad, Pakistan. His research interests and expertise include homomorphic encryption, attribute-based encryption, searchable encryption, blockchain, and privacy-preserving machine learning. His professional services include a technical program committee member and a reviewer of several international journals and conferences.

SHAHZAIB TAHIR (Senior Member, IEEE) received the M.S. degree in information security from NUST, Pakistan, and the Ph.D. degree in information engineering from the City, University of London, U.K. He was a Research Fellow with the City, University of London, U.K. He is currently an Assistant Professor with the Department of Information Security, NUST, and also the Founder and the CTO of CityDefend Ltd., U.K. His research interests include applied cryptography and cloud security. He is also an Alumni of Innovate U.K. CyberASAP. His professional services include a technical program committee member and a reviewer of several international journals and conferences.

ASIF IQBAL (Member, IEEE) received the B.S. degree in telecommunication engineering from NUCES-FAST, Pakistan, in 2008, the M.S. degree in wireless communications from LTH, Lunds University, Sweden, in 2011, and the Ph.D. degree in electrical and electronics engineering from The University of Melbourne, Melbourne, Australia, in 2019. He is currently a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. Previously, he was a Faculty Member of NUCES-FAST as an Assistant Professor. His research interests include signal processing, deep learning, sparse signal representations, and privacy-preserving machine learning.



SYED AZIZ SHAH is currently an Associate Professor in mobile health with the Healthcare Technology and Innovation Theme, Research Centre for Intelligent Healthcare, Coventry University, U.K. He has authored or coauthored more than 80 technical articles in top-rank peer-reviewed multi-disciplinary journals. His research interests include multiple disciplines, including wireless sensing, machine learning, cyber security, intelligent healthcare technologies, mobile health, prototype design, radar sensing for healthcare technologies, physiological measurements, machine learning, and cyber security for intelligent healthcare. He has served as a TPC member, a special issues editor, and a reviewer of several international conferences and journals.

ABDULLAH BAZ (Senior Member, IEEE) received the B.Sc. degree in electrical and computer engineering from UQU, in 2002, the M.Sc. degree in electrical and computer engineering from KAU, in 2007, and the M.Sc. degree in communication and signal processing and the Ph.D. degree in computer system design from Newcastle University, in 2009 and 2014, respectively. He was the Vice-Dean and the Dean of the Deanship of Scientific Research with UQU, from 2014 to 2020. Currently, he is an Associate Professor with the Computer and Network Engineering Department. He was the Vice Dean of the Deanship of Human Resource, the General Director of the Decision Support Center, the General Director of the Data Management Office, and the Technical Consultant of the University Vice Chancellor with UQU. His research interests include data science, ML, AI, VLSI design, EDA/CAD tools, intelligent transportation, computer system and architecture, smart systems, and smart health. He is a member of the IEEE CAS/FWSC Standards Committee and an Associate Editor of the IEEE International Midwest Symposium on Circuits and Systems (MWSCAS). Since 2015, he has been served as a Review Committee Member for the IEEE International Symposium on Circuits and Systems (ISCAS) and a member of the Technical Committee for the IEEE VLSI Systems and Applications. He served as a Reviewer in a number of journals, including the IEEE INTERNET OF THINGS JOURNAL, the *IET Computer Vision*, the *Artificial Intelligence Review*, IEEE ACCESS, and the *IET Circuits, Devices & Systems*.