Privacy Preserving Machine Learning Using Homomorphic Encryption

MCS

By

Aftab Akram

A thesis submitted to the faculty of Information Security Department, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment of the requirements for the degree of MS in Information Security.

Dec 2022

Thesis Acceptance Certificate

Certified that final copy of MS Thesis written by **Mr. Aftab Akram**, Registration No. **00000318233**, of **Military College of Signals** has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations/MS Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members and local evaluators of the scholar have also been incorporated in the said thesis.

Signature: ______________________________

Name of Supervisor **Asst. Prof. Dr. Fawad Khan**

Date: ______________________________

Signature (HOD): ______________________________

Date: ______________________________

Signature (Dean/Principal) ____________________

Date: ______________________________

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Aftab Akram

Dec 2022

Dedication

This thesis is dedicated to my Family and my Supervisor for their endless support, and encouragement throughout my research work.

Acknowledgement

All praises to Allah Almighty for the strength He gave me in completing this thesis. I would like to convey my gratitude to my supervisor Asst. Prof. Dr. Fawad Khan and co-supervisor Asst. Prof. Dr. Shahzaib Tahir, for their supervision and constant support. I would also like to thank my committee members; Assoc. Prof. Dr. Muhammad Faisal Amjad and Maj. Zeeshan Zulkifl for their support, guidance and knowledge regarding this topic. Their comments and suggestions throughout the thesis work are major contributions to the success of this research.

Last, but not the least, I am highly thankful to my father and mother. They have always stood by my dreams and aspirations and have been a great source of inspiration for me. I would like to thank them for all their care, love and support through my times of stress and excitement.

Abstract

Neural network-based machine learning algorithms have shown outstanding results and are currently being widely used in numerous fields. These machine learning algorithms demands considerable computing power for internal calculations and training with big datasets in a reasonable amount of time. In recent years, clouds provide services to facilitate this process, but it introduces new security threats, as the machine learning algorithms mainly rely on the utilization of personal data for training and classification which frequently has privacy implications. To overcome this problem, we propose new approach for operating deep neural networks on encrypted data. Homomorphic encryption is a cryptographic technique, which allows to perform computations on encrypted data, but it also has some limitations associated with it. However, it only supports limited number of addition and multiplication operations in encrypted domain. Existing works only cater simple machine learning algorithms like binary classifiers and simple neural networks in the encrypted domain. Moreover, these simple machine learning algorithms does not provide the required accuracies and also handle a limited number of datasets. To address these issues deeper neural networks are required, which on the other hand increases the computational complexity. In this study, we create novel methods for implementing deep neural networks within the realistic limitations of homomorphic encryption techniques. We mainly concentrate on convolutional neural networks for training and encrypted classification. To begin, we provide techniques for approximating the activation functions typically employed in CNNs (e.g., ReLU and Sigmoid) with low degree polynomials, which is required for efficient homomorphic encryption schemes. The models are then trained using approximation polynomials rather than the original activation functions, and their performance is evaluated. In the end, we apply convolutional neural networks to encrypted data for privacy preserving classification by varying the various Homomorphic encryption scheme's parameters and evaluate the model performance. The proposed scheme ensures privacy while attaining the maximum accuracy.

Table of Contents

List of Figures

List of Tables

Notations

evk	evaluation key
pk	public key
sk	secret key
λ	security parameter
c	ciphertext
$\mathbb{Z}$	set of integers
N	set of natural numbers
B	set of vectors
$\mathcal{L}$	represent lattice
b_i	basis vectors
R	set of real numbers
$\|\boldsymbol{v}\|$	norm of a vector
$< a, b >$	dot product of two vectors
T	plaintext modulus
N	polynomial modulus degree
Q	coefficient modulus
E	noise (error)
N_m	highest Degree polynomial (m)

Chapter 1

Introduction

1.1 Overview and Motivation

Cloud computing empowers anyone–from individuals to enterprises–to migrate on premise workloads to a third-party. Cloud computing might not be feasible in all circumstances, despite the fact that it can significantly enhance the computer power available to customers with low budgets and processing capacity. For instance, the GDPR [1] regulation of the European Union (EU) governs the privacy and data protection laws within EU and limits the information that may be transmitted beyond the EU. The GDPR may be violated, if the computation of sensitive data is outsourced to a cloud provider located outside the EU. One method for maintaining data privacy and GDPR compliance is to make encryption of data before sharing it to a cloud server. But what if someone want to perform computations on the data at cloud server? Homomorphic encryption (HE) helps us to perform such kind of computations. It allows the users to execute calculations on encrypted data at outsource environment while ensuring the cloud server wouldn't learn anything about the data of the user.

Although the use case is straightforward, implementing this kind of solution is a difficult task. The biggest challenge is creating the homomorphic version of the function that has to be evaluated later. First, the input data must be encoded as homomorphic plaintexts and this encoding will impact the efficiency of the resulting circuit. Second, HE schemes only support basic arithmetic operations like addition and multiplication and it cannot support high-level functions like rounding, evaluating non-polynomial functions. Finally, multiplicative depth of evaluation function has a direct relation with HE parameters, higher depth functions demand bigger HE parameters, which decreases the efficiency of the scheme and increases computing time.

For years, the security and machine learning scientists have been fascinated by learning the model without having access to raw material. Ideally, we want the confidential data to be encrypted before storing it on cloud and certain data analysis performed without ever decrypting the data. HE is a strong candidate for secure outsourced data computations, however because of the aforementioned problems, implementing real-world machine learning tasks in an outsourced privacy-preserved environment is quite difficult. Existing solutions [2][3][5] can only handle simplified low-depth circuits such as logistic regression and simple neural network and these learning algorithms only handle datasets in a texture format. The primary objective of this thesis is to provide the practical support to high depth real-world machine learning model like deeper neural network. Deeper neural network also allows us to handle datasets comprises of textural as well as image format and also the model accuracy of such networks are far better than the simple learning algorithms. This work will examine the performance of the purposed model by using real-world datasets and demonstrate its feasibility in different sectors like health, genomes analysis and business analytics etc.

1.2 Problem Statement

Technology advancement in current era introduces a new sort of currency in everyday human experience known as individual’s privacy. Machine learning, as a technical front-runner that plays a leading role in many current developments, 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 required a significant computing resource in order to analyze enormous volumes of data efficiently. Taking advantage of cloud resources is one answer to this dilemma. When it comes to security, a cloud-based solution invites a number of problems. On the other hand, what if it was possible to incorporate the best aspects of both worlds, i.e., utilizing cloud resources while maintaining individual security, while performing machine learning on cloud settings?

Supposing a previously trained machine learning model is stored on the cloud. A client encrypts his data and sends it to cloud server. The model uses encrypted data to process a result that can only be decrypted by the individual. This identical scenario has been demonstrated achievable through the use of privacy-preserving classification. Therefore, it is crucial to comprehend how encryption is used inside privacy-preserving classification when evaluating the confidentiality and effectiveness of a system. The core idea of privacy-preserving classification problem is to use the encrypted data to make encrypted predictions. The design and implementation of a prediction model involves the use of three datasets: training, validation, and testing. A training dataset is used during the learning phase to determine the weights that comprise the predictive model. During the learning phase, a validation dataset is used to fine-tune the model's architecture and meta-parameters, as well as to query the model's performance on unseen data. After the learning process, the final model's predictive ability is verified using a testing dataset. This is known as the classification process.

The learning phase of privacy-preserving classification uses unencrypted datasets, whereas the inference phase uses encrypted datasets. It looks like a client-server architecture as shown in Fig 1. In this scenario, the prediction model has already been trained on cloud server, yet the cloud server would like to change it so that it can classify inputs that have been encrypted. So, learning phase is straightforward and uses unencrypted training and validation datasets to update weights and fine-tune model architecture. But during the classification phase, the distinction can be seen, where the model provides an encrypted prediction on encrypted testing dataset. The proposed solution to the classification problem in a privacy preserved environment is based on homomorphic encryption.

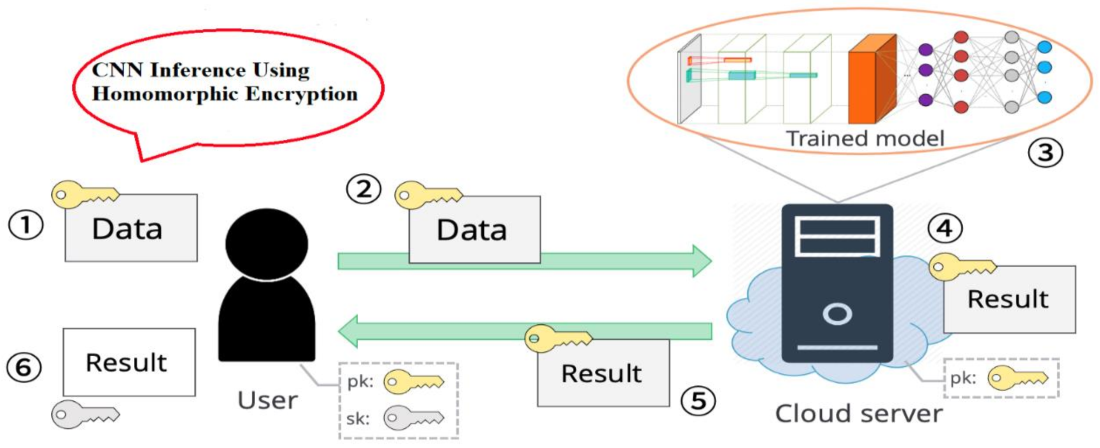


Figure 1. *HE-based Privacy Preserving Classification*

1.3 Research Objectives

The main objectives of thesis are discussed as follows:

- To ensure the privacy of user's data while performing computation on the data in out-sourced environments e.g., cloud server.
- Application of HE for higher depth multiplicative circuits like Convolutional Neural Network (CNN).
- Assessing efficient approximation techniques for nonlinear computation functions to make them compatible with HE.
- Verification of proposed solutions using the MNIST [6] dataset.

1.4 Thesis Contribution

In literature, the privacy-preserving classification problem has several solutions that are based on HE [2][3][5][4]. While these solutions claim to be accurate and efficient, they are mainly unexplored and understudied. In fact, finding even a straightforward case study is challenging due to the inadequate documentation and missing source codes. Also, most of these works only employ the HE for simple machine learning algorithms like simple logistic regression etc. To employ the HE for deep learning algorithm, like convolutional neural network (CNN), is a challenging task. This study focuses on a more thorough investigation of combining HE with CNN.

This study combine the CNN with HE using an open source Microsoft cryptographic library called SEAL[7], which is built on the BFV [8][9] and CKKS [10] schemes. Although both encrypted training as well as encrypted classification are conceivable, but 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 (ReLU & 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 were correct. Following the success of the preliminary results, a bigger seven-layer network is built to conduct encrypted classification on handwritten digit dataset, named as MNIST [6] dataset. 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, the HE-parameters are adjusted to observe how parameter size impacts efficiency and accuracy in order to investigate the behavior of privacy-preserving classification in the prospective of cryptography/security.

1.5 Research Methodology

The research work starts from literature review of the existing proposed schemes being used for privacy preserved machine learning using HE. The literature review is done from various academic sources. This research then narrows down to privacy preserved CNN classifications problems using HE while listing down the drawbacks of existing schemes and formulates the problem. Then, it discusses the necessary changes needed in CNN layer to make it compatible with HE schemes for encrypted classification purposes in later part of the thesis.

The implementation is done using Python version of SEAL [11], called Pyfhel [12], in VS Code in a Windows-based environment. Several modules are also integrated for the complete implementation, which will be discussed in detail in relevant chapter. In the end, a road map for future research areas in the privacy preserving classification will be discussed. Fig 2 represents the major highlights of the research methodology.

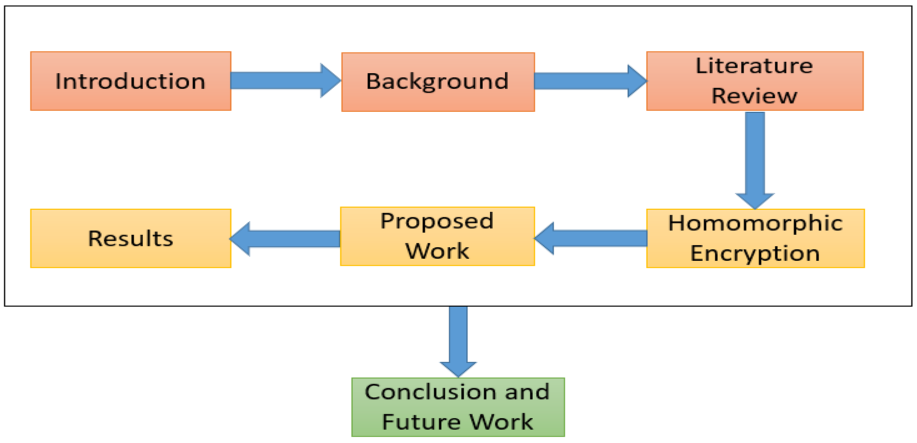


Figure 2. Research Methodology

1.6 Thesis Organization

The thesis is organized as follows:

a. Chapter 2 contains a brief mathematical background related to post-quantum cryptography in the context of homomorphic encryption.
b. Chapter 3 contains a brief literature review of HE-based privacy preserving machine learning solutions. Some well-known privacy preserving techniques are also discussed in this chapter.
c. Chapter 4 elaborates the main homomorphic encryption scheme used in our proposed model.
d. Chapter 5 introduces the proposed work. The practical feasibility of the proposed model is also discussed in this chapter. Privacy of different components of proposed CNN model is also discussed.
e. Chapter 6 covers the details of implementation of the proposed work in python-based environment. MNIST dataset is used to provide a comparative study. The results are generated in the form of graphs and are also presented in this chapter.
f. Chapter 7 marks the end of this document, concluding the results and some future recommendations to cater for the issues faced during this thesis.

Chapter 2

Preliminary Studies

This chapter provides the necessary mathematical background along with some basics of cryptography. It also includes the most critical definitions for comprehending the FHE schemes and definitions. At the end some basics of machine learning algorithms, especially the convolutional neural network, are also discussed.

2.1 Introduction

The term cryptology originates from the Greek terms *kryptós* and *logos*, which means "hidden word." Generally, cryptology is a science that studies how to hide confidential information. Cryptography and cryptanalysis are two complementing branches in cryptology, with cryptography being the science of building secure ciphers and cryptanalysis being the science of cracking ciphers. This thesis will concentrate on cryptography, specifically encryption schemes along with their daily life applications. The goal of cryptography is to hide confidential information from unauthorized parties, by providing some among the properties like Confidentiality, Integrity, Non-repudiation and Authentication. Cryptographic algorithms are based on the concept of computational hardness, which makes them practically tough to break by an adversary. The cryptosystems are techniques and protocols which meet all or some of the characteristics listed above.

Encryption refers to the process of encrypting a message or piece of information so that only authorized people may access it, ensuring confidentiality. Symmetric (Private-key) and Asymmetric (Public-key) are two types of encryption schemes. Asymmetric cryptosystems have separate encryption and decryption keys, whereas symmetric cryptosystems use the same key for both. The fact that symmetric encryption is faster than asymmetric encryption is one of its advantages. However, one disadvantage is that the key must be exchanged securely. In order to understand the concept of HE, we only focus on the public key cryptography in this chapter.

2.2 Public-Key Encryption

Public-key encryption (PKE) enables the users to transmit messages privately without using a shared secret [13][14]. Cryptographic algorithms, which depends upon one-way functions (OWF) are used to generate public and private keys. In PKE, anyone having public key can encrypt the message and create a ciphertext. However, only those who have the access of associated private key can able to decrypt the ciphertext.

A PKE scheme mainly comprises the following three algorithms [14]:

Key Generation. $KeyGen(1^{\lambda}) \rightarrow (pk, sk)$: *It generates private* $(\boldsymbol{pk})$ *and public* $(\boldsymbol{sk})$ *keys while taking security parameter* λ.

Encryption. $Enc(pk, m) \rightarrow c$: *It encrypts message* m *using* $\boldsymbol{pk}$ *and generates ciphertext* c.

Decryption. $Dec(sk, c) \rightarrow m$: *It decrypts ciphertext* c *by utilizing the* $\boldsymbol{sk}$ *to recover the message* $m \in M$.

Cryptographic algorithms are built on the assumption of computational hardness, making them difficult to break in practice by any adversary with sufficient knowledge. The security given by a particular encryption is determined by its average-case hardness, rather than the effort required to crack it in the worst-case scenario. The desired hardness is either a proof that minimal number of steps required to obtain the solution is extremely large and thus impossible to break, or a reduction to an NP-Hard problem under assumption $P \neq NP$. The hardness problems of most public key encryption systems are depended upon integer-factorization and discrete-logarithm problems.

2.3 Homomorphic Encryption

In traditional encryption, it requires to decrypt a message in order to perform any kind of operations on it. In contrast, HE allows to conduct computations directly on the ciphertext. While decrypting the ciphertext, the resulting plaintext will match the result of performing the computation on the corresponding plaintext. A homomorphic encryption scheme with encryption algorithm E over an operation'$*$' supports the following equation:

$$E(m_1) * E(m_2) = E(m_1 * m_2);\ \forall\, m_1;\ m_2 \in M$$

where M is the messages space [15].

In contrast to PKE, the HE schemes mainly contains the following four algorithms:

Key Generation. *$KeyGen(1^{\lambda}) \rightarrow (pk, sk)$: It generates private* ***(pk)*** *and public* ***(sk)*** *keys while taking security parameter λ.*

Encryption. *$Enc(pk, m) \rightarrow c$: It encrypts message m using* $\boldsymbol{pk}$ *and generates ciphertext c.*

Decryption. *$Dec(sk, c) \rightarrow$ µ: It decrypts ciphertext c by utilizing the* $\boldsymbol{sk}$ *to recover the message $m \in M$.*

Evaluation. *$\boldsymbol{Eval}(C, (c_1, \dots, c_l), \boldsymbol{pk}) \rightarrow \boldsymbol{c'}$: It takes public key* $\boldsymbol{pk}$ *and applies a circuit $C: \mathcal{C}^l \rightarrow \mathcal{C}$ to $c_1, \dots, c_l$, and outputs a ciphertext c'.*

For the ciphertext to be successfully decrypted following an evaluation process, its format must be maintained. Furthermore, the size of ciphertext increases after every operation is applied on it. To perform infinite operations, there is a need to keep size of ciphertext within specific limit. The bound of number of operations on ciphertext classifies the HE schemes into three categories: PHE, SHE and FHE as shown in Fig 3. The detail of these schemes is given in the following sections.

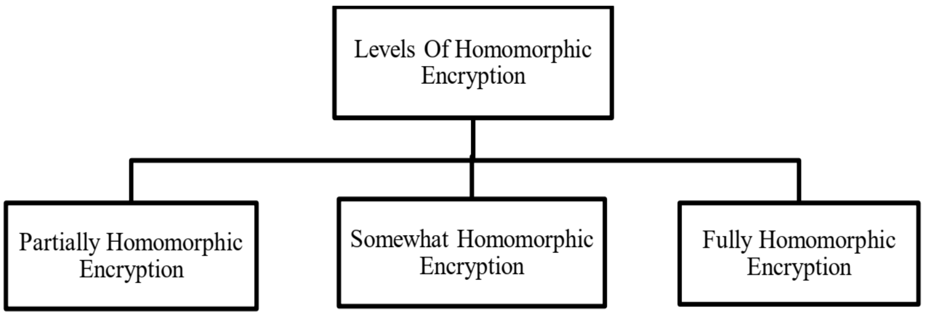

Levels Of Homomorphic Encryption
Partially Homomorphic Encryption
Somewhat Homomorphic Encryption
Fully Homomorphic Encryption

Figure 3. *Homomorphic Encryption Types*

2.3.1 Partially Homomorphic Encryption

There are many conventional cryptosystems that only allow to perform a single operation on ciphertext. These are classified as PHE.

PHE: It is either an additive homomorphic scheme that allows only additive operations on encrypted data or a multiplicative homomorphic scheme that allows only multiplicative operations on encrypted data.

Partially homomorphic cryptosystems include RSA [13], ElGamal [16], , Benaloh [17], Paillier [18], Goldwasser-Micali [19] and Damgård-Jurik [20] and we will now show the homomorphic properties for unpadded RSA and Paillier.

RSA: *The RSA* [13][14] *scheme contains three algorithms as (KeyGen, Enc, Dec), define as follows:*

- $\boldsymbol{KeyGen}(\lambda) \rightarrow (\boldsymbol{pk}, \boldsymbol{sk})$:
  1. Generate two distinct primes p and q, compute $N = pq$ and $\varphi(N)$.
  2. Take an integer $e \overset{R}{\rightarrow} \mathbb{Z}_{\varphi(N)}$ such that $GCD(e, \varphi(N)) = 1$, then compute its (modular) inverse $d = e - 1 \; mod \; \varphi(N)$.
  3. Set: $\boldsymbol{pk} = (N, e)$ and $\boldsymbol{sk} = (N, d)$
- $\boldsymbol{Enc}(\boldsymbol{pk}, \boldsymbol{m}) \rightarrow \boldsymbol{c}$: *Compute $\boldsymbol{c} = \boldsymbol{m^e} \; mod \; \boldsymbol{N}$.*
- $\boldsymbol{Dec}(\boldsymbol{sk}, \boldsymbol{m}) \rightarrow \boldsymbol{m}$: *Compute $\boldsymbol{m} = \boldsymbol{m^d} \; mod \; \boldsymbol{N}$.*

RSA provides the homomorphic property w.r.t multiplicative, which means that one can multiply two ciphertext to receive multiplication of the underlying plaintexts. Let's utilize the same key for encrypting two messages, m_1 and m_2:

$$\boldsymbol{Enc}(m_1) = m_1^e \, mod \, N$$

$$\boldsymbol{Enc}(m_2) = m_2^e \, mod \, N$$

The homomorphism is then defined as follows:

$$\boldsymbol{Enc}(m_1) \cdot \boldsymbol{Enc}(m_2) = m_1^e m_2^e \, mod \, N = (m_1 m_2)^e mod \, N = \boldsymbol{Enc}(m_1 \cdot m_2)$$

Paillier: *The Paillier* [18] *scheme also comprises three algorithms (KeyGen, Enc, Dec), define as follows:*

- $\boldsymbol{KeyGen}(\lambda) \rightarrow (\boldsymbol{pk}, \boldsymbol{sk})$:
  1. Generate two distinct primes p and q, such that $GCD\big(pq, (p-1)(q-1)\big) = 1$. Compute $n = pq$ and $\lambda = lcm(p-1, q-1)$.
  2. Take an integer $g, u \in \mathbb{Z}_{n^2}^*$ such that $GCD(n, L(g^\lambda (mod \; n^2)\,)) = 1$, and $L(u) = (u-1)/n$.
  3. Set: $\boldsymbol{pk} = (n, g)$ and $\boldsymbol{sk} = (p, q)$

- $\boldsymbol{Enc(pk,m)} \to \boldsymbol{c}$: *Generate a random value* $r \epsilon \mathbb{Z}_{n^2}^*$ *and compute* $c = g^m r^n \, mod \, n^2$.
- $\boldsymbol{Dec(sk,m)} \to \boldsymbol{m}$: *Compute* $m = \left(\frac{L(c^\lambda \, mod \, n^2)}{L(g^m \, mod \, n^2)}\right) mod \, n^2$

Paillier has additive homomorphic property. Let us use the same key to encrypt two messages, m_1 and m_2*:*

$$\boldsymbol{Enc}(m_1) = (g^{m_1} r_1^n \, mod \, n^2)$$

$$\boldsymbol{Enc}(m_2) = (g^{m_2} r_2^n \, mod \, n^2)$$

The homomorphism is then defined as follows:

$$\boldsymbol{Enc}(m_1) \cdot \boldsymbol{Enc}(m_2) = (g^{m_1} r_1^n \, mod \, n^2)(g^{m_2} r_2^n \, mod \, n^2)$$
$$= g^{m_1+m_2}(r_1 r_2)^n \, mod \, n^2 = \boldsymbol{Enc}(m_1 + m_2)$$

2.3.2 Somewhat Homomorphic Encryption

Depending upon the number of operations it can perform, HE schemes can be categorized as somewhat homomorphic scheme.

SHE. It can allow to perform both the operations of multiplicative and additive, but allows limited number of repetitions. This restriction is defined by the scheme's ability to correctly decrypt ciphertext associated with homomorphic operations.

All HE schemes that had been presented up to 2005 could only perform one operation either addition or multiplication. Boneh-Goh-Nissim (BGN) [21] is the first SHE schemes which allows only one multiplication operation but infinite number of additions, while keeping the size of ciphertext constant. Another example of SHE schemes is BFV [9] scheme. In general, the ciphertext of such homomorphic encryption scheme has a noise parameter, and the noise must be less than a certain limit in order to be decrypted correctly. This scheme can perform additive and multiplicative homomorphism on encrypted data, but after each operation the noise level increases in the generated homomorphic ciphertext. So, it can only perform a limited number of operations in order to keep the noise parameter as small as possible.

2.3.3 Fully Homomorphic Encryption Schemes

In 2009, the first FHE scheme was introduced by Craig Gentry in his doctoral thesis [19]. Based on his work, other researchers also tried to develop their own practical FHE schemes.

FHE. It can allow to execute infinite number of operations, both multiplication as well as addition, on encrypted data [22].

Although Gentry's FHE scheme looked promising, but its high computational cost made it impractical. In order to make his scheme practical for real-world applications, several improvements had been made. Though efforts to develop new FHE techniques continued, the majority of them were centered on lattices problems. Depending upon the hardness problems, FHE schemes can be divided into four types as follows:

1. **Ideal lattice-based:** Firstly, Gentry [23] proposed Ideal lattice-based FHE scheme, then other researchers improve his work, Smart and Vercauteren [24].
2. **NTRU-based:** NTRUEncrypt is an encryption scheme which has homomorphic properties [25].
3. **RLWE-based:** Vaikuntanathan and Brakerski [26] proposed a FHE schem based upon RLWE problem.
4. **Integers-based:** An approximate-GCD problem based FHE scheme proposed by Van Dijket al [27].

2.4 Post-Quantum Cryptography

The security of various encryption systems is currently jeopardized due to the threat of quantum computing. Shor created a quantum algorithm in 1994 that can solve the integer factorization as well as discrete logarithm problems [28]. This means that with the presence of quantum-based computers, all cryptographic schemes based on these assumptions will not achieve the same level of security. ElGamal and RSA are two of the schemes affected. Because these are used to protect many types of sensitive data, breaking them could have serious consequences for privacy and security. As a consequence, cryptographers must develop new protocols based on completely novel concepts and assumptions. The following are some possible research directions that are thought to be quantum secure:

- Code-based Cryptography
- Multivariate-based Cryptography
- Lattice-based Cryptography
- Hash-based Cryptography

Table 1 lists some classical cryptosystems as well as their current security status w.r.t quantum computers. In this chapter, we look at lattice-based cryptography, which is thought to be resistant to quantum computers, meaning no one has yet discovered a way to break it. Also, most of the practical FHE schemes also have their hardness based on lattices.

***Table 1.** Security of Classical Cryptosystem in Quantum Era*

Cryptosystem	Broken by Quantum Algorithms?
RSA [13]	Not Secure
Diffie-Hellman [29]	Not Secure
Elliptic curve [30], [31]	Not Secure
McEliece [32]	Secure
NTRU [33]	Secure
Lattice-based [34]	Secure

2.4.1 Lattice-based Cryptography

Lattice-based encryption looks to be the most favorable options for post-quantum-based cryptography. The lattice-based cryptographic long-term security may be guaranteed for two main reasons. First, it has been established that many lattice-theory problems are NP-Hard [35]. Second, the worst to average case reductions are applicable to these lattice problems [34]. This means that picking any random instance of the problem will be as hard as solving the worst case.

2.4.2 Lattices

A lattice is a collection of points in n-dimensional space that has a periodic structure [36]. Let $\boldsymbol{B} = \{\boldsymbol{b_1}, \dots, \boldsymbol{b_n}\}$ be a set of *n*-linearly independent vectors in $\mathbb{R}^m$hen set of all integer linear combinations of the vectors in $\boldsymbol{B}$ will be the lattice which is generated by $\boldsymbol{B}$: $\mathcal{L}(\boldsymbol{B}) = \{\sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z}\}$ [36][37][38]. This provides the definition [38]:

$$\mathcal{L}(B) = \{\vec{x} \times \boldsymbol{B} : \vec{x} \in \mathbb{Z}^n\}$$

For n-linearly independent vector having dimension $n = m$ is defined as full rank lattice. Hence, the following definition [38]:

$$\boldsymbol{B} = \{\boldsymbol{b_1}, \dots, \boldsymbol{b_n}\}, \quad \boldsymbol{b_i} \in \mathbb{R}^n$$

An example of a lattice in $\mathbb{R}^2$is shown in Fig 4.

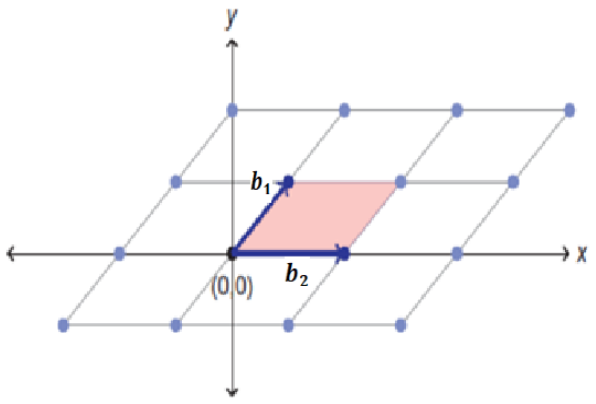


Figure 4. *A lattice in* $\mathbb{R}^2$

Ideal lattice, which have additional structure than ordinary lattices, particularly the structure of an ideal, was the foundation of Gentry's method [19]. Lattices have a group structure, but ideal lattices, as the name implies, have an ideal structure. Some problems in lattice-based cryptography are easily solved by using bases of a specific structure. We define a bad basis as one in which solving a specific lattice problem is often no simpler than on a random basis. Good bases are ones in which a certain problem may be solved easily. The public key is a "bad" basis whereas the secret key is a "good" basis for lattice-based FHE algorithms [39]. A good bass is often composed of vectors that are short and nearly orthogonal [39].

2.4.3 Lattice Problems

This section discusses several typical hard problems that serve as the basis for a variety of lattice-based cryptographic schemes. The problems involve finding the shortest and closest vectors in a lattice.

2.4.3.1 SVP & CVP

The Shortest Vector Problem (SVP) has been extensively studied, and it appears to be intractable in general, even with quantum algorithms. The SVP seeks a nonzero vector, often known as a short or shortest vector, whose Euclidean norm is the smallest among all other nonzero lattice vectors. This is considered simple in a two-dimensional lattice, but it becomes difficult to solve in multiple dimensions.

Definition 1 *(SVP). Given an arbitrary basis* $\boldsymbol{B}$ *for an* n*-dimensional lattice* $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$*, compute a non-zero vector* $\boldsymbol{v} \in \mathcal{L}$*, such that* $\|\boldsymbol{v}\| = \boldsymbol{\lambda}_1$. [40][37][41]

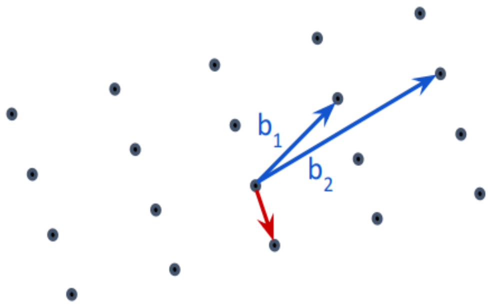


Figure 5*. Shortest Vector Problem*

CVP is the generalized form of SVP. Previous research work [39] [52] has concluded that CVP is as much hard as SVP. CVP asks for a vector which is not too far from a specific target point, and it doesn't necessarily have to be the closest one.

Definition 2 *(CVP). Provided any arbitrary lattice basis* $\boldsymbol{B}$ *for an* n*-dimensional lattice for some target point* $\boldsymbol{t} \in \mathbb{R}^n$*, compute* $\boldsymbol{v} \in \mathcal{L}$ *such that* $\|\boldsymbol{t} - \boldsymbol{v}\|$ *is minimal.*[40][42]

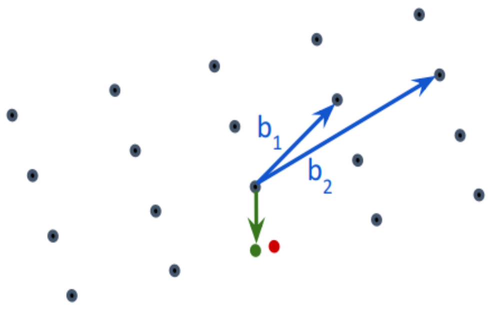


Figure 6*. Closest Vector Problem*

There are two major distinctions between SVP and CVP. First, the SVP requires a lattice point near zero, whereas CVP asks a lattice point near an arbitrary point in space.

Second, in CVP the solution can be an all zero vector, whereas in SVP it cannot. As a result, it is impossible to use CVP to solve SVP by obtaining the shortest vector that is close to the origin because doing so would result in the zero vector. The SVP and CVP problem are shown in above Fig 5 and 6, respectively.

2.4.3.2 Approximate SVP & CVP

Due to the hardness of solving SVP and CVP, cryptographers have considered approximation versions of these problems, which are particularly suitable to cryptography. Approximation algorithms only yield answers that are confirmed within a certain factor γ of the optimum.

In approximate-SVP (SVP_γ), the task is to identify a non-zero vector located at a distance of no more than $\gamma\lambda_1(\mathcal{L})$, for $\gamma= \gamma(n) \geq 1$.

Definition 3 *(SVP_γ). Fix $\gamma > 1$. Given any arbitrary basis $\boldsymbol{B}$ for n-dimensional lattice $\mathcal{L}$, compute a non-zero vector $\boldsymbol{v} \in \mathcal{L}$ such that* $\|\boldsymbol{v}\| = \gamma\, \boldsymbol{\lambda}_1$. [37][40]

In an approximation-CVP (CVP_γ), this involves to find a lattice vector at most distance of γ, for $\gamma= \gamma(n) \geq 1$.

Definition 4 *(CVP_γ). Fix $\gamma > 1$. Given any arbitrary basis $\boldsymbol{B}$ for an n-dimensional lattice $\mathcal{L}$ and some target point $t \in \mathbb{R}^n$, compute $\boldsymbol{v} \in \mathcal{L}$ such that* $\|\boldsymbol{t}-\boldsymbol{v}\| \leq \gamma\, \|\boldsymbol{t}-\boldsymbol{xB}\|$. [40]

The following problems are the decision variant of approximating the shortest vector and closest vector in a given lattice within a factor γ.

Definition 5 *(Gap Shortest Vector Problem (Gap SVP_γ)). Given* **B** *and r, decide whether $\lambda(B) \leq r$ or if $\lambda(B) \geq \gamma.r$ (Instances where $r < \lambda(B) < \gamma.r$ are not considered.)* [37]

Definition 6 *(Gap Closest Vector Problem (Ga SVP_γ)). Given $\boldsymbol{B}, x \in \mathbb{R}^n$ and r, decide whether dist$(x, L) \leq r$, or if $dist(x, L) \geq \gamma.r$. (Instances between r and $\gamma.r$ are not considered.)* [37]

Variants of this approximation problem are commonly used to demonstrate the security of cryptosystems [43].

2.4.4 Learning With Error

Regev [44] introduced the Learning With Errors (LWE) problem in 2005, which is a generalization of the Learning Parity with Noise (LPN) problem. The LPN problem is the same as decoding random linear codes. It is a well-studied problem which is believe to be hard [44]. Regev demonstrated that his public-key cryptosystem based on LWE hardness was significantly more efficient than other suggested public-key cryptosystems based on unique-SVP, a subset of SVP. He further proves LWE's hardness via a quantum reduction from the worst-case lattice problem SVP, where a quantum reduction is a reduction that employs quantum computing. As a result, it has become an essential building block in modern cryptographic systems, as well as a prominent topic in current research. The problem with LWE is that it is inherently inefficient owing to a quadratic overhead. In LWE, the size of public key is $O(mnlog\, q) = \tilde{O}(n^2)$. Additionally, it increases the message size by a factor of $O(n\, log\, q) = \tilde{O}(n)$ by each encryption [44].

Definition 7 *(Decisional Learning With Errors (DLWE)). Let* n *and* q *be positive integers, and* χ *an error distribution over* $\mathbb{Z}$*. Let* $\boldsymbol{s}$ *be a uniformly random vector in* $\mathbb{Z}_q^n$*. The DLWE is to distinguish* $A_{s,\chi}$ *from the uniform distribution* U *from* m *independent samples* $(\boldsymbol{a_i}, b_i) \epsilon\, \mathbb{Z}_q^n \times \mathbb{Z}_q$ *where every sample is distributed according to either:* $A_{s,\chi}$ *or the uniform distribution.* [44][37]

Definition 8 *(Search Learning With Errors). Let* n *and* q *be positive integers, and* χ *an error distribution over* $\mathbb{Z}$*. Let* $\boldsymbol{s}$ *be a uniformly random vector in* $\mathbb{Z}_q^n$*. The search LWE is to find* s *from* m *independent samples*$(\boldsymbol{a_i}, b_i) \epsilon\, \mathbb{Z}_q^n \times \mathbb{Z}_q$ *drawn from* $A_{s,\chi}$*.* [44][37]

2.4.5 Ring Learning With Error

To address LWE's inefficiency, Regev, Peikert, and Lyubashevsky devised Ring Learning With Errors (RLWE) problem [45]. In most cases, n noisy LWE equations may be replaced by a single noisy RLWE equation, which obviously increases efficiency. RLWE, defined as LWE over ideal lattice, is an algebraic variant of LWE. These are more structured than random lattices. It can be interpreted mathematically as replacing the group $\mathbb{Z}_q^n$ with the ring $\mathbb{Z}_q[x]/(x^n + 1)$. [44][45]

***Definition* 9** *(RLWE). Consider the ring* $\mathbb{R} = \mathbb{Z}_q[x]/(x^n + 1)$ *with* n *as power of* $\mathbf{2}$ *and an error distribution* χ *over* R. *Let* $\boldsymbol{s}$ *be uniformly random sampled from* $\mathbb{R}_q$. *The decision RLWE is to distinguish* $A_{s,\chi}$ *from the uniform distribution* $\mathbb{R}$ *from* m *independent samples* $(\boldsymbol{a_i}, b_i) \in \mathbb{R}_q \times \mathbb{R}_q$ *where every sample is distributed according to either:* $A_{s,\chi}$ *or the uniform distribution.* [45][37]

2.5 Neural Network

A network of neurons organized in the layers is referred to as a neural network (NN). Every neuron receives an input, process it through a function, and then outputs the outcome of function. The layer that the neuron belongs to, determines the structure of this function. Input, hidden, and output layers are the three layers that make up a simple neural network [46]. The structures of the neurons in each layer vary. Input layer neurons only receive input; their outputs are identical to the input values. In hidden layer the neuron has an input vector $(x_0, \dots, x_n)$, a weight vector $(w_0, \dots, w_n)$, and an output y. This formula is used to compute the output.

$$y = f(\sum_{i=0}^{n} x_i * w_i)$$

In the above equation, f is an activation function. Several functions might be utilized as activation functions such as step, hyperbolic, ReLU and sigmoid functions. The output layer is the last. This layer's neurons are sometimes simple (like those in the input layer) and sometimes complicated (like in hidden layers). Each layer contains bias neuron that is linked to all the neurons in subsequent layer. This work only focuses on the fully connected feed-forward neural network. In such kind of networks all neurons of each layer are linked to all neuron present in subsequent layer. A simple neural network is shown in the Fig 7 below.

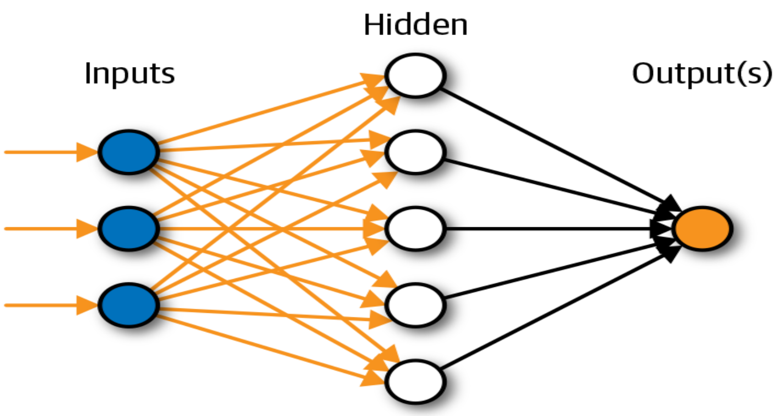


Figure 7. *A Neural Network*

Convolutional Neural Network (CNN), a generalized form of neural networks, is discussed in the following section.

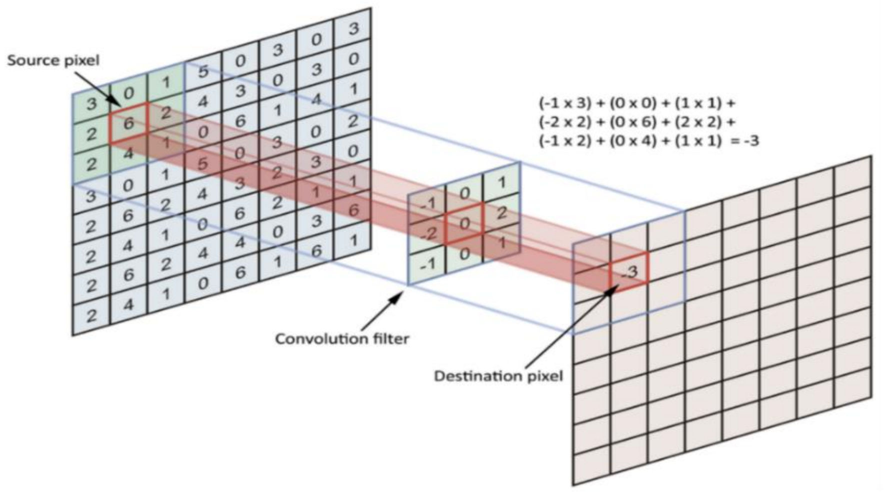
2.6 Convolutional Neural Networks (CNNs / Conv-Nets)

In machine learning CNNs [47] are certain kind of feed-forward type neural network which are very useful in different areas specially in image recognition and image classifications. CNNs are built up of cascading layers that accept image data as input and transform it into label scores as output. Layers include in CNNs are as following:

- Convolutional Layer
- Activation Layer
- Pooling Layer
- Fully-Connected Layer

2.6.1 Convolutional Layer

The convolutional layer contains a sliding filter that is applied to the input image. When a filter is applied in an image it extracts 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 number of weights that are learnt 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., stride of (2, 2) means that at each step a filter is moving two units to the left or down. This layer calculates dot-product between filter weights and associated values in pixel's neighbor by convolving the pixels in the image. This step just requires two operations, one is addition and second is multiplication, so one can apply the same procedure in the encrypted domain easily. Fig 8 illustrates the convolution process [48].



Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel

Figure 8. *Convolutional Layer*

2.6.2 Activation Layer

In CNN there is another layer which contains a non-linear function, is known as 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. Some of the activation functions used in practice are shown in Fig 9.

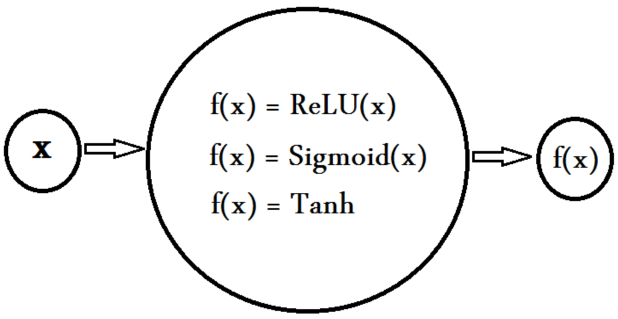


Figure 9. Activation Functions

The following equations represent the sigmoid and ReLU functions.

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

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

Since it is obvious that these equations involve nonlinear functions, we must change them in order to make them compatible with HE schemes.

2.6.3 Pooling Layer

Pooling-layer is used to decrease the size of the data by sub-sampling it. This is also a non-linear layer. This layer usually comes after the activation layer. The two most common non-linear pooling layers i.e., Average and Max pooling layers, are shown in Fig 10 [49]. The maximum value inside the subsection is output of Max pooling layer, and average of all values inside the subsection is the output of the Average pooling layer.

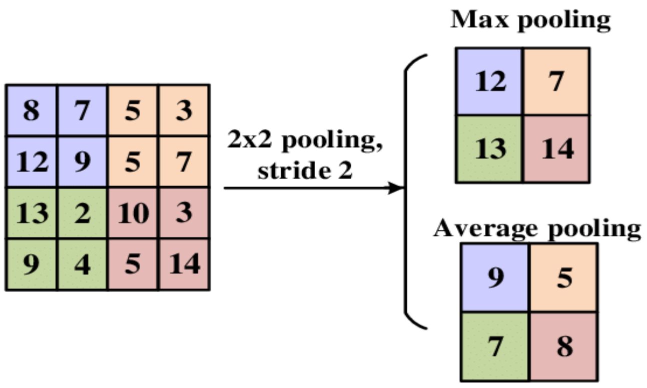


Figure 10. *Pooling Layers*

2.6.4 Fully Connected Layer

It is usually the last layer of CNN. All neurons in this layer are linked to all other neuron of previous layer, as seen in Fig 11 [50]. The total sum of weights in this layer is the product of total number of neurons in the preceding and current layers. This layer's output is how many classes are included in the dataset.

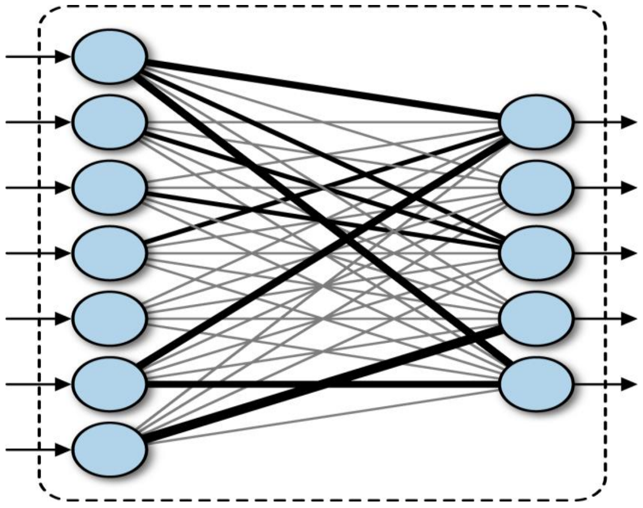


Figure 11. Fully Connected Layer

Chapter 3

Literature Review

In this chapter, first we introduce some well-known techniques that are currently used to perform machine learning in privacy-preserving environment. The next step is to undertake a thorough literature review of earlier studies in the perspective of privacy-preserving machine learning. We largely concentrated on homomorphic encryption-based solutions from the techniques mentioned above.

3.1 Privacy Preservation Techniques

Prime objective of this research work is centered around performing the privacy preserved classification using convolutional neural network (CNN). So, the method chosen for this purpose must not only allow for secure computing but also give a high level of security. Secure Remote Computation (SRC), Homomorphic Encryption (HE), and Multi-Party Computation (MPC) are three common privacy preservation strategies that were investigated in order to secure CNN.

3.1.1 Secure Hardware (Intel SGX)

The Secure Remote Computation (SRC) problem refers to the ability of an individual to run software on some remote computer while maintaining a level of security [51]. According to this problem, remote computer must be hosted by an untrustworthy party, emphasizing the need of data confidentiality along with data integrity.

The SRC problem is solved by Intel SGX, with the goal of securing user-level programs through the use protected memory sections (enclaves). An individual uploads his data into a protected enclaves and performs the private calculations using a particular set of CPU instructions. In contrast to other secure hardware systems, Intel SGX solely makes use of attestation to validate contents in the protected enclave. Initially, Intel SGX considered as the appropriate solution of SRC problem, however subsequent investigations have revealed certain flaws in it. In fact, studies shows that the Intel SGX has different vulnerabilities, particularly it is vulnerable to cache timing-attacks [52]. Furthermore, sources indicate that the security assurances of Intel SGX are not apply to cloud settings [51][52]. Because the goal of this study is to improve cloud security, secure hardware has been ruled out [51].

3.1.2 Secure Multi-Party Computation

Secure Multi-Party Computation (SMPC) is an approach which allows for distributed computation of functions. The primary objective of MPC is to offer a system which allows different parties to collaborate on specific computations while respecting the privacy of their input data [53]. Each participant in MPC provides their own input data, which is converted into smaller chunks and sent to other servers each masked with a random value. This is how MPC allows for cooperative computing while preserving the privacy of each person's personal data [54]. Although, MPC has a little computational overhead benefit, but it demands several rounds of communication between the individuals participating in the protocol.

3.1.3 Homomorphic Encryption

HE is a distinct type of encryption scheme that allows to performs direct calculation on ciphertext. Similar to many other PKE cryptosystems, to encrypt some data, HE cryptosystem also uses public-key while the decryption is done using private-key. However, after the data has been encrypted using the public key, HE also permits normal arithmetic operations (additions and multiplications) to be applied to the encrypted data. For instance, if you add two encrypted values together using homomorphic addition, the outcome will be the plaintext values added in an encrypted manner, as illustrated in Fig 12. It means, operations performed in the ciphertext space thus resemble operations performed in the plaintext space.

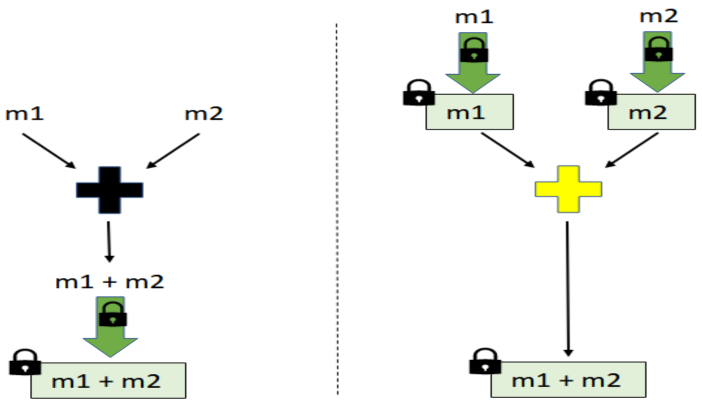


Figure 12. Homomorphic Addition

3.2 Related work

Graepel et al. [55] trained two machine learning classifiers using a Somewhat HE schemes. These algorithms included Linear Mean and Fisher's Linear Discriminate. In order to circumvent HE algorithms limitations, they introduced division-free algorithms. They did not take into account more complex algorithms, 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. [56] proposed privacy-preserving classification model for three distinct machine learning algorithms named as Naive Bayes, Hyper-plane Decision and Decision trees. The author combined garbled circuits with three homomorphic encryption schemes named as Piallier, BGV and Quadratic Residuosity schemes. They use 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. [57] examined theoretical elements of constructing neural networks in the encrypted domain using polynomial approximation. Dowlin et al. [3] extended this work by presenting CryptoNets. It was a first detailed studied CNN classifier for encrypted data. The author employed Microsoft SEAL, a levelled homomorphic encryption technique that supported SIMD. The author used 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, since they did not support the exponential function. They trained the given model using unencrypted data, then utilized it to classify encrypted data. On the MNIST dataset, they got an overall accuracy of 98.95%. This CryptoNets was able to process 48068 cases per hour. The accuracy of CryptoNets was improved in a study by Chabanne et al.[4] by combining the solution's original concepts with batch-normalization approach. They employed ReLU as the activation function in their scheme. They utilized a mix of Taylor series along with batch-normalization for ReLU activation function approximation.

Jiang et al [58] proposed a privacy preserving deep learning model named as 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 used CNN with a square activation function, two fully connected and one convolutional layers.

To train a simple neural network in privacy preserved environment Phong et al. [59] suggested a technique based on additive homomorphic encryption. The author pointed out a weakness in Shokri et al. [60] work, that leaked client data during training process. The main concept was to allow a server to upgrade the model (learning) by aggregating user gradient values.

Hesamifard et al. [5] proposed a work named as 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. [61] proposed a privacy preserving technique for CNN training as well as classification purposes. Each activation layer was preceded by 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. So, they modified the CNN with these changings to make it compatible with HE.

Juvekar et al. [62] proposed a framework named as Gazelle, which combined HE with MPC, for privacy preserving classification purpose. The goal of this study was to retain the model privacy in the server and to make it simpler for client to perform a classification without exposing his input data to 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 privacy. The experiment demonstrates that, in terms of runtime, Gazelle completely surpasses other well-liked methods like MiniONN [63] and Cryptonets [3].

Sanyal et al. [64] proposed a framework, TAPAS, which used encrypted data to speed up parallel processing in privacy preserved enjoinment. They tried to overcome the lengthy process of classification in context of HE. The key contribution was to develop a novel approach to accelerate binary computing in Binary Neural Network (BNN). All data are initially converted into binary by the algorithm. After that, it performs an XNOR operation between encrypted and unencrypted data to compute the inner product. They then count how many 1’s there are in the preceding step's outcome. They next determine if the difference between the bias and the number of bits was greater than twice the amount that was counted. If the answer was yes, they gave the activation function a value of 1, and if the answer was no, they gave it a value of -1. They also decreased the evaluation step time by evaluating the gates in parallel processing at the same level. Another work of Bourse et al. [67] proposed a technique named as 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 liner from with respect to depth of neural network. Their neural network contained the discretized value of weights and biases as compared to standard neural network. They employed sign function as the activation function in their network. To update the output of the neuron, they employed the bootstrapping for computing the activation function. They successfully demonstrated that by increasing network size, BNN might achieve accuracy that was comparable to that of normal neural network.

Table 2: *Comparative Analysis of exiting privacy preserving*

Study	HE Scheme	ML Technique	Dataset	Run Time (s)	Accuracy (%)	PoC/PoM Considerations	Comments
Graepel et al. [55]	BFV	Linear Mean & Fisher's Linear Discriminate	Breast Cancer	255.7	95.00	PoC	It only supports simple ML algorithm.
Bost et al. [56]	Piallier, BGV Quadratic Residuosity	Hyperplane Decision, Naïve Bayes & Decision Trees	Breast Cancer & ECG	14.77	---	PoC	It only supports simple ML algorithm.
Dowlin et al. [3]	YASHE	CNN	MNIST	697	98.95	PoC	They use Taylor series for function approximation.
Chabanne et al. [4]	BGV	CNN	MNIST	---	99.30	PoC	Their crypto parameters are not clear.
Jiang et al. [58]	CKKS	CNN	MNIST	28.59	98.10	Both	It only caters simple NN, missing pooling layer.
Phong et al. [59]	Additive-HE Piallier, LWE-based	CNN	MNIST & Speech	120	97.00	Both	It only handle simple 3-layer NN.
Hesamifard et al. [5]	BGV	CNN	MNIST	320	99.52	PoC	It can classify many instances for each prediction round.
Liu et al. [61]	BGV	CNN	MNIST	477.6	98.97	PoC	Their CNN not include pooling layer.
Juvekar et al. [62]	HE + MPC	CNN	MNIST & CIFAR-10	----	----	Both	It combines CNN with HE and MPC.
Sanyal et al. [64]	TFHE	BNN	Cancer & MNIST	147	98.60	Both	It works only for simple binary NN such as BNN.
Bourse et al. [67]	TFHE	DiNN	MNIST	1.64	96.35	Both	DiNN is another form of BNN.
Brutzkus et al. [68]	CKKS	CNN	MNIST & CIFAR-10	----	98.95	PoC	Their crypto parameters are not clear.
Lee et al. [69]	RNS-CKKS	CNN ResNet-20	CIFAR-10	14694	98.43	PoC	They employ bootstrapping after Conv. and ReLU.

Chapter 4

Homomorphic Encryption Scheme

4.1 Introduction

Homomorphic encryptions can perform computations on ciphertext directly. However, due to the properties of the various HE variants, not all are suitable for all tasks. SHE is a variant of HE in which one can perform many encrypted operations sequentially, but the total number of encrypted operations is limited by the scheme's initialization parameters.

As mentioned above, we have made use of the SEAL implementation, a popular homomorphic library for usage in higher-level applications. The library is developed in C++ and has a wrapper for languages like C#. Its source code is available on GitHub under the open-source MIT license. One of the main schemes that is implemented in SEAL is the Fan-Vercauteren (FV) scheme [9], is discussed in [53] along with some improvements. FV is a SHE schemes and its security relies on the RLWE problem, a quantum-secure problem with high security. SEAL also includes the CKKS scheme [10] in addition to the FV scheme.

4.2 Description of the FV Scheme

The FV technique is the homomorphic encryption scheme employed by the Microsoft SEAL library, and it is based on the algebraic ring structure. Basically, algebraic rings are mathematical sets of elements inside a modulus that enable the binary operations addition and multiplication. To make the FV scheme work, our initial plaintext numbers, the ones we wish to decrypt, must be obtained in the ring structure $\mathbb{R}_t$. The ring $\mathbb{R}_t$ is define as $R_t = \mathbb{Z}_t[x]/x^n + 1$, which includes only those integer number from $\mathbb{Z}$ for which there exists a polynomial having degree less than n with coefficients reduced modulo t. Here, the scheme is initialized by defining the important initialization parameters of plaintext modulus t, ciphertext modulus q, and degree of polynomial modulus n. 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 our original numbers must be a member of the ring structure $\mathbb{R}_t$ in order to be encrypt able under this scheme so, we first encode each one to make it a member of the ring structure. Any number, whether it be an integer or a rational number, must be encoded into a plaintext polynomial in $\mathbb{R}_t$ before it can be encrypted under the scheme, according to the ring $\mathbb{R}_t$. After the appropriate integers have been encoded into $\mathbb{R}_t$ they are encrypted into a ciphertext array of at least two polynomials in the ring structure $\mathbb{R}_q$ where q is the coefficient modulus and is specified as an encryption parameter before the encryption occurs. Setting the initialization parameters is covered in section 4.4 below.

This section includes detailed explanations of the encryption and decryption processes used in the FV scheme to ensure its correctness. A $a \overset{\$}{\leftarrow} \mathbb{R}_2$ denotes that a is sampled uniformly from the finite set $\mathbb{R}_2$. The scheme's main algorithms are as follows:

- Generate public keys *pk*, secret keys *sk* and evaluation keys *evk* using the algorithms *PublicKeyGen*, *SecretKeyGen* and *EvaluationKeyGen*.
- *Eneryption(pk,m)*: Let public key $pk = (p_0, p_1)$ and message $m \in \mathbb{R}_t$. Sample $e_1, e_2 \leftarrow \chi$ and $u \overset{\$}{\leftarrow} \mathbb{R}_2$. The ciphertext ct is given as $ct = \left([\Delta m + p_0 u + e_1]_q, [p_1 u + e_2]_q\right)$.
- *Decryption(sk,ct)*: Let $s = sk$, $c_0 = ct[0]$ and $c_1 = ct[1]$. Compute $m' = \left[\left\lfloor \frac{t}{q}[c_0 + c_1 s]_q \right\rceil\right]_t$ to get the decryption of m inti m'.

4.2.1 Key generation

Private and public key pair is used throughout the encryption process to transform a plaintext number into a ciphertext number. Two stages are adopted for key generation. First stage is to generate private-key (s). The process of generating the private-key involves creating an n-term random polynomial. Furthermore, a uniform sample of each coefficient is taken from a set of $\{-1,0,1\}$. Next, public-key (pk) is generated by first taking another temporary random polynomial (called a polynomial) from the ciphertext space, i.e., a polynomial having its coefficients modulo the q variable. The coefficients are equally sampled over the whole q range. The temporary random polynomial a will have the same n terms as the secret key. The next step is to create a random error polynomial called e for the public key. To achieve this, we sample n coefficients from a discrete Gaussian distribution with values that are significantly lower than q. Afterwards, the two polynomials ($pk = [-as + e]\ and\ pk = a$) define the public key in the following manner:

$$pk = ([-as + e], a)$$

Now that the s and pk keys have been generated, we may execute encryptions. So, the next thing we'll do is examine how the encrypt-process-decrypt procedure handles the encryption step.

4.2.2 Encryption

Recall that a plaintext polynomial is changed into a pair of ciphertext polynomials throughout the encryption process. When there are n terms in the plaintext polynomial, each with a coefficient modulo t. Additionally, the ciphertext polynomial pair contains n terms with coefficients that are modulo q. We will need to create three additional little polynomials, identical to those used to in public key generation to conduct the encryption. Two error polynomials e_1 and e_2 will be constructed from identical discrete Gaussian distribution which was used to generate e in public key. Along with e_1 and e_2 we will produce the third polynomial u whose coefficients will be uniformly sampled from the same set as the secret key, namely the set $\{-1,0,1\}$. Following the generation of the three polynomials e_1, $e_2\ and\ u$ the two ciphertext polynomials are determined as follows:

$$ct = \left(\left[pk_0.u + e_1 + \left\lfloor\frac{q}{t}\right\rfloor.m\right]_q, [pk_1.u + e_2]_q\right)$$

The ciphertext ct computed above correctly hides our message m in the combination of random noise values. Because our initial message m is a plaintext polynomial with modulus t variable coefficients, it is scaled up first by $[\frac{q}{t}]$ and then hidden by summing with $(pk_0.u + e)$. Despite the fact that the e_1 is taken as a sample from a discrete Gaussian distribution, the term $pk_0.u$ effectively masks our message, making it difficult to distinguish from random noise. The reason why the same plaintext message will always generate a different ciphertext is due to $pk_0.u$.

We may identify five components of ciphertext by further analyzing the computations for each single encryption step. These includes private key, public key, message, noise and mask. The mathematical expansion of the encryption phase shows the five encryption components as follows:

$$ct = \left(\left[\overbrace{- \underbrace{a\,u}_{mask} \quad \underbrace{s}_{secret}}^{pk_0} + \underbrace{e\,u + e_1}_{noise} + \underbrace{\left\lfloor \frac{q}{t} \right\rfloor . m}_{message} \right], \left[\underbrace{\overbrace{a}^{pk_1} \;. u}_{mask} + \underbrace{e_2}_{noise} \right] \right)$$

4.2.3 Decryption

After understanding the encryption, now we move to understand the decryption process. To decrypt a ciphertext ct using the FV technique, we first remove the masking by summing the ciphertext's two polynomials to yield the polynomial shown below:

$$[ct_0 + ct_1 . s]_q = \left[-aus + e_0 u + e_1 + \left\lfloor \frac{q}{t} \right\rfloor . m \right]_q + [a . u + e_2]_q . s$$

$$= \left[e_2 s + e_0 u + e_1 + \left\lfloor \frac{q}{t} \right\rfloor . m \right]_q$$

The above expansion demonstrates that in addition to our message m scaled by $[\frac{q}{t}]$, additional information known as the inherent noise, v, is present in the ciphertext. The equation above provides a definition for this inherent noise as

$$v = [e_2 s + e_0 u + e_1]_q$$

We next compute by scaling the ct polynomial back to the values in modulo t in order to ensure that the decryption is successful. Meanwhile, the noise terms v will be removed by rounding off. The noise terms must be small enough to be rounded off in order for this to succeed; otherwise, the decryption will fail. This scaling down step is accomplished by first multiplying with $\frac{t}{q}$, and then rounding off the little noise terms as follows:

$$m' = \left[\left\lfloor \frac{t}{q} \left[e_2 s + e_0 u + e_1 + \left\lfloor \frac{q}{t} \right\rfloor . m \right]_q \right\rceil \right]_t$$

On the other hand, we may write this by emphasizing the noise polynomial as:

$$m' = \left[\left\lfloor \frac{t}{q} \left[v + \left\lfloor \frac{q}{t} \right\rfloor . m \right]_q \right\rceil \right]_t$$

The plaintext message m from the previous equation is decrypted to its corresponding plaintext message m'. If no operation was done on the ciphertext, then m' = m; otherwise, m' will represent the outcome of the operation.

The noise polynomials represented by v must have coefficients that are small enough to be rounded off and scaled down by $[\frac{t}{q}]$. In contrast, if the noise coefficients are larger, they will quietly create an inaccurate result since they will end up closer to a different integer than their intended one. This finding implies that the liberty to manage an equivalent quantity of noise is provided by the difference in $[\frac{q}{t}]$. The amount of noise that may be tolerated during the decryption process grows along size of the difference between the q and the t.

4.3 Noise Budget (Circuit Depth)

Each ciphertext can only handle a certain number of homomorphic operations. SEAL refers to this restriction as the noise budget, whereas other researchers in the homomorphic encryption community refer to it as the circuit depth. We will also use the terminology "noise budget." This noise budget reduces towards zero as we do homomorphic operations. When the noise budget reaches its zero limit, all homomorphic operations produce garbage values because the coefficients of the polynomial representing the ciphertext exceed the coefficient modulus q, an encryption parameter. As a result, the decryption method will be unable to decipher the ciphertext within the encryption parameters that have been specified. The most significant consideration is the noise budget. Because, it is the noise budget, which permits or prevents a computing party from performing additional homomorphic operations on ciphertext.

It is important to note that if we combine a ciphertext with a zero or low noise budget with another ciphertext that has an adequate noise budget during an arithmetic operation, the noise budget for the resulting ciphertext will be zero. This makes it clear that the output won't be successfully decrypted and decoded if one of the ciphertext operands has inadequate noise budget.

As stated in the SEAL documentation [22], the initial noise v, in a ciphertext is calculated using the formula below:

$$v_i = \frac{q \; mod \; t}{q} . \|m\| . N_m + \frac{7nt}{q} . \min noiseMaxDeviation, \;\; 6 \times noiseStandaredDeviation$$

In the formula for calculating the initial noise budget above, we have our original message as m, the encryption parameters as n, t, q, and the highest degree of polynomial m as N_m. The random noise distribution is defined by the standard deviation and the maximum deviation of the sample. The initial noise formula shows us that initial noise budget for the identical message m is dictated by the initialization parameters of the encryption scheme t, q and n. The next section defines these parameters.

4.4 Parameter Selection $(\boldsymbol{t}, \boldsymbol{q}, \boldsymbol{n})$

The encryption initialization parameters have a substantial impact on the homomorphic processes. The initialization settings have an impact on the actual encryption /decryption, along with the performance and outcome of the operations. These settings must be configured before any integers are encrypted or homomorphic processes are performed. The security keys (the public/private and evaluation keys) of the scheme are created based on these encryption settings. The following are the three primary encryption parameters:

4.4.1 Plaintext Modulus $(\boldsymbol{t})$

The plaintext (coefficient) modulus, which specifies the maximum size of the plaintext data that may be encrypted, can be any positive integer. It has significant effects on the noise budget's initial value in a newly encrypted ciphertext and how much of it is used up during homomorphic multiplications. For a good performance without impacting the noise budget, the t value must be kept as low as feasible [7].

4.4.2 Ciphertext Modulus ($\boldsymbol{q}$)

The FV scheme's ciphertext (coefficient) modulus is a product of one or more tiny prime integers. The magnitude of the coefficient modulus should be considered a key component in defining the noise budget. To be correctly decoded, a ciphertext's noise value should be less than the q value. The decryption method will fail to decipher ciphertext with a noise value greater than the q value. A high coefficient modulus must be utilized, if a big noise budget is necessary for complex computations. However, studies have shown that a higher coefficient modulus q also reduces the scheme's level of security. By simultaneously raising the polynomial modulus n while increasing q, this decrease in security level can be regained [7].

When we discuss the coefficient modulus's size, we are referring to the bit length of its product, which can be one or more smaller prime values. The coefficient modulus in SEAL is a positive composite number that is the sum of several primes with a maximum bit size of 60-bits.

The size of the polynomial modulus n and the number of prime elements in the coefficient modulus have the greatest influence on performance. Thus, based on experiments, it is recommend using as few factors in the coefficient modulus is possible for good performance.

4.4.3 Polynomial Modulus ($\boldsymbol{n}$)

The polynomial modulus n is the maximum value that can be used in a polynomial to represent a plaintext or a ciphertext. The value n should be thought of as mainly affecting the security level of scheme. The HE schemes becomes more secure as the polynomial modulus increases. To properly encode integers into the ring R, the value of n must be the power-of-2 cyclotomic polynomial, i.e., $(1.x^{(power\ of\ 2)} + 1\ )$. Because there are more coefficients in ciphertext due to a larger polynomial modulus n, all operations become slower as a result. Based on security and efficiency considerations, the SEAL documentation suggests that n takes values of $1024, 2048, 4096, 8192, 16384$ or 32768 for typical computation scenarios.

4.5 Relinearization

Multiplications in the FV and other related homomorphic encryption schemes increase the number of polynomials in the ciphertext. Relinearization is a technique for lowering the number of polynomials to an acceptable level in order to control noise growth.

Relinearization is required for a number of reasons. These include the fact that processing a larger polynomial than a smaller one takes longer time. In order to obtain the output, convolutional neural networks use operations like the multiplication of several numbers over multiple layers, which is an extremely computationally expensive algorithm. Another intriguing justification for using relinearization that we discovered throughout our study is that smaller ciphertext results in a lesser increase in noise. Relinearization can be used in the CNN after each multiplication to reduce noise, which is depending on the size of the ciphertext operands. This can be seen by a simulation of the noise growth when two ciphertext are multiplied together, one with an increasing number of polynomials and the other with a fixed polynomial count size of 2. The SEAL documentation [7] for multiplication includes a relinearization formula that may be used to determine the noise in the output of the multiplication.

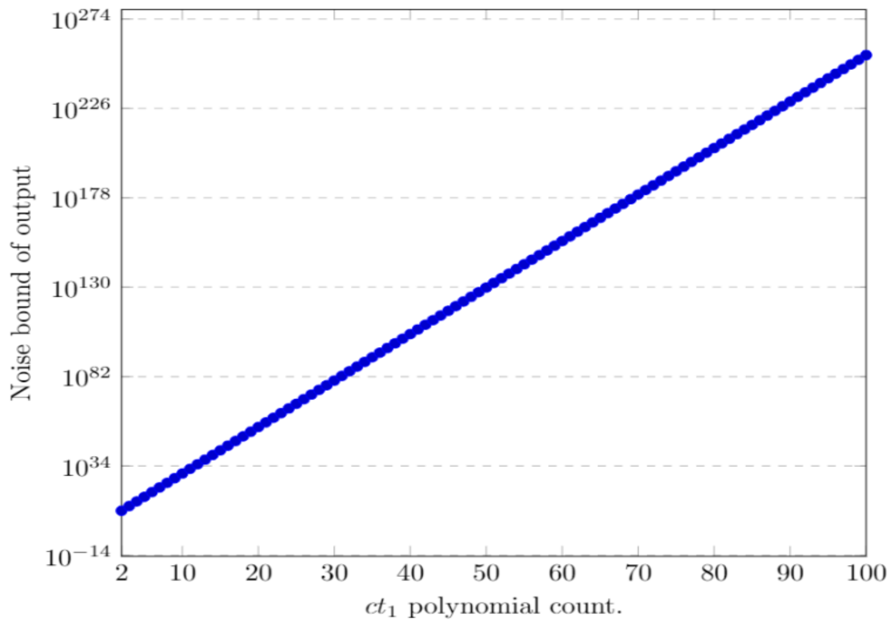


Figure 13. *Effect of ciphertext polynomial size on noise budget*

We get the conclusion from this simulation that noise increases exponentially as ciphertext size increases.

4.6 Number encoding

In SEAL, the numbers that we want to compute must be encoded in a polynomial of the type $x^n + 1$. Here, the polynomial's coefficient is x, and n is a power of two. The polynomial modulus, which was discussed above, is represented by $x^n + 1$. The SEAL library encodes integers and fractions in a somewhat different fashion, as explained below.

4.6.1 Integer Encoding

We give an example to explain integer encoding. Let the encoding base $x = 2$, then the integer $30 = 2^4 + 2^3 + 2^2 + 2^1$ is encoded in polynomial form as $1.x^4 + 1.x^3 + 1.x^2 + 1.x^1$. Similarly, for encoding base $x = 3$, the integer $30 = 2^4 + 2^3 + 2^2 + 2^1$ encoding as a polynomial is $1.x^3 + 1.x^1$.

4.6.2 Fractional Encoding

Fixed-precision rational numbers are used to implement fractional encoding, with the integral part handled identically to integer encoding and the fractional part handled slightly differently. It extends the number in a specified base x, possibly truncating an infinite fractional portion to finite precision. For example,

$$30.75 = 2^4 + 2^3 + 2^2 + 2^1 + 2^{-1} + 2^{-2}$$

Here the encoding base is $x = 2$. For the sake of understanding, let the polynomial modulus is $(1.x^{1024} + 1)$. The integer part of the above fractional number is encoding as the same way as encoding an integer, but the fractional part is transferred to the highest degree part of the polynomial with the coefficient signs changed. Because we are working with ring structures that can only contain positive integers, the negative coefficients are always encoded as a residual of the plaintext coefficient modulus t. In our example, for $t = 6$ and $n = 1024$, the fractional encoding of the number $30.75 = 2^4 + 2^3 + 2^2 + 2^1 + 2^{-1} + 2^{-2}$ is given as

$$30.75 = 5.x^{1023} + 5.x^{1022} + 1x^4 + 1x^3 + 1x^2 + 1x^1$$

4.7 HE Coded Libraries

Over the years, various authors have released a number of homomorphic encryption libraries, most of which are intended for specific implementations. Different libraries include different HE schemes. Table 3 lists the most popular libraries along with the corresponding HE schemes that each library offers.

Table 3. *HE Libraries*

Library/Scheme	BFV	BGV	TFHE	FHEW	CKKS
SEAL	✓	✓			✓
HElib		✓			✓
PALISADE	✓	✓			✓
cuHE			✓		
TFHE-Chimera	✓		✓	✓	✓
FHEW				✓	
HEAAN					✓

Chapter 5

Proposed Work

Cloud environment provide the ease to access data and use on-demand resource sharing from anywhere 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. 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.

This chapter first describes the system model and the entities involved in it. Then, it provides the threat model which describes the potential threat in the given system model. It’s also discussing the privacy of each component of network in the proposed system model in the context of threat model. At the end, the necessary modifications needed in CNN layers to make them compatible with HE are discussed. These modifications allow to perform privacy preserved classification at outsourced environment.

5.1 System and Threat Model

HE helps the clients in outsourcing their critical data securely. During the outsourcing of the data, confidentiality and privacy of client data can be compromised by the malicious or curious server. System model helps in determining the entities involved as well as about the functionality of the proposed protocol. Threat modelling helps in finding potential exploits which can later become stern threats. Moreover, it also helps in securing the data from the entities like internal threat actors or external malicious threats. The threat model works with our system model and helps in establishing the effectiveness of our proposed model.

5.1.1 System Model

In order to maintain the privacy of the major components of the privacy preserving classification model as a service framework, we take into account the system model shown in Fig 14, in which the cloud server uses a 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.

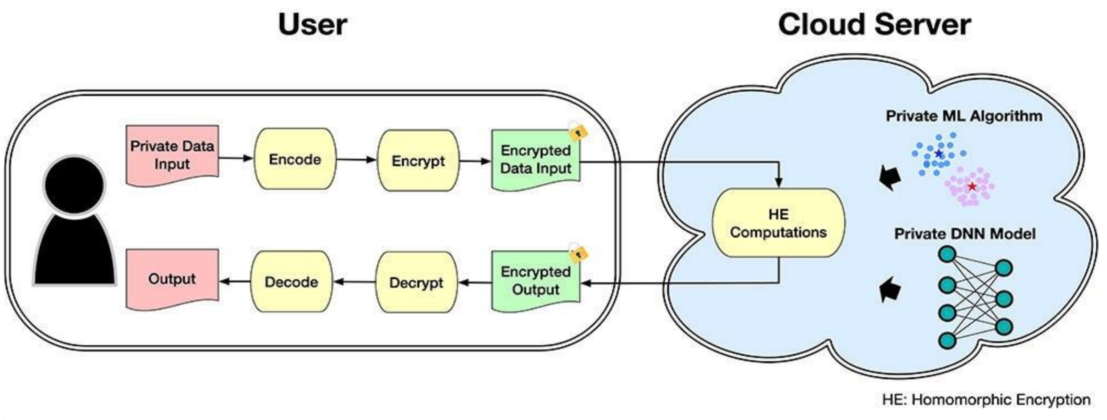


Figure 14. *System Model*

In this system model, the server already has the training dataset and it builds a model from this plain dataset. The final model is in plaintext form placed at cloud. The client gives encrypted instances to cloud server which classifies these instances and returns the encrypted results to client. Intent in this system model is to protect the privacy of feature values of inputs and prediction of unseen instances against server, and also to protect the privacy of machine learning algorithm and model parameters against the client.

5.1.2 Threat Model

To create a secure protocol, we must identify potential threats and attacks that could target a system. Adversaries frequently aim is to jeopardize security requirements such as confidentiality, integrity, and availability. During threat modelling, it is critical to understand these threats and attacks, as well as their implications for security. Our system model of privacy preserving classification typically involves two entities, one is data owner (client) and second is cloud server and it involves a two-way communication i.e., from client to server and from server to client. In first case, the client encrypts the data and generates the required keys for homomorphic calculations, and sends encrypted queries and the public key parameters to cloud server. In this case, from the client’s viewpoint, the main threat in the system is either the cloud server or any eavesdropper. As the queries are encrypted so eavesdropper would not get any meaning full information. As far as the server is concerned, we assume it as semi-honest threat model in which cloud server strictly follows the protocol specifications but may passively collect transmitted inputs and try to infer useful information about client's data. In second case, cloud performs desired computations on an encrypted query and sends encrypted results to client. In this case, from the server's prospective, the main threat in the system is either the client or any eavesdropper. As the results are encrypted so eavesdropper would not get any meaning full information. As far as the client side is concerned, we also assume it as semi-honest threat entity in which the client strictly follows the specifications of protocol but also try to infer useful information about machine learning algorithm and the weights of trained model placed on server. The adversial matrix of our model is shown in Table 4.

Table 4. *Adversial Matrix*

	Known	**Unknown**
Client	• Input data • Data types of input of the neural network • Data types of outputs of the neural network • Encryption and decryption keys	• Machine Learning Algorithm • Model Parameters
Server	• Machine Learning Algorithm • Model Parameters	• Input data • Encryption and decryption keys

5.2 Model's Components Privacy Considerations

The privacy of different parts of a classification model must be taken into account when performing privacy-preserved classification in an outsourced environment. These components include feature values, predictions for unseen instances, ML algorithm and ML model privacy. In the sub-sections, we will discuss the privacy of each component individually and explain how our proposed model will ensure the privacy of these components.

5.2.1 Feature Values

From the perspective of the data owner, feature values are one of the most important considerations. As these values include sensitive information of data owner, exposing them to the cloud server would be a serious security issue in the system. Medical records are an example of sensitive data that is stored as feature values. The Health Insurance Portability and Accountability Act (HIPPA) standard ensures that the data of the patient is kept private. This parameter is used to assess the proposed protocol's security. A non-authorized party should only receive minimal (or no) information about the feature values. In our suggested model, the data owner encrypts the feature values before sending them to the cloud. As a result, no non-authorized party receives any information.

5.2.2 Predictions of Unseen Instances.

The classifier's output is another component which should be consider during privacy preserving classification. All private classification protocol considers this component to be the client's private information. For example, the outcome of evaluating a patient's medical data contains sensitive information regarding the patient's present health state. Any entity other than the data owner does not have access to the result of the classifier. The classifier results are encrypted in our proposed protocol, so server has no access to them, ensuring the privacy of unseen instances.

5.2.3 Machine Learning Algorithm

If parties do not share the ML algorithm, then the learning algorithm is equally crucial. The privacy of this component is taken into account in privacy preserving classification scenarios, and it should be considered throughout the protocol design process. Consider a company that specializes in data analysis for other organizations. One of their assets is their data processing approach, and they don't disclose how the model is constructed. After the analysis, they just send the results to the client. The server's privacy requires that privacy of this component be protected. The privacy of this component is maintained in our proposed protocol since client receives no information from the cloud server regarding the steps involved in data analysis.

5.2.4 Model Privacy

The model privacy is essential for client as well as the server. Actually, the model includes the patterns of the dataset as well as information gained from instances, so it is critical for the client. As a result, the model should not be made accessible to the server. From the perspective of the server, it is a server asset, similar to the analysis algorithm. Suppose a company provides a classification service to the clients in a privacy preserved environment. In this case, the model must be kept secret from the client because now it is a server asset. As a result, maintaining the privacy of model is also important. The model is built by the server and not transferred to the client in our protocol; as a result, the model's privacy is protected from the client.

5.3 CNN Layer Design

The primary purpose of this research work is solely to perform classification on encrypted data. So, the CNN layers are designed while considering feed-forward network only and back-propagation phase is omitted. The only operations supported by SEAL is addition and multiplications, so CNN layers are designed while keeping these limitations in mind. In our CNN model, the activation layers contain the non-linear functions, thus the primary challenge is to deal with this function in our model. To combat this challenge, different polynomial approximation techniques are explored e.g., Numerical approximation method, Taylor series and Chebyshev approximation. By doing analysis it is found that Chebyshev approximation technique is best for approximating the activations functions. More detail of approximating functions is given in Chapter 6. Before creating a privacy-preserving CNN, all the layers of CNN are researched, implemented and then tested both in plaintext as well as ciphertext space. While performing computations using SEAL, in case of plaintext the input and output vectors have data type of long while in encrypted case (ciphertext space) these vectors have data type of ciphertext. The plain layers of CNN are used as a reference for comparison to verify the accuracy of the encrypted classification.

5.3.1 Activation Functions Design

The activation function takes a 1-D/3-D vector as input, and the outputs a 1-D/3-D vector. The output vector has type long or ciphertext depends on whether an input to the function is encrypted or unencrypted. In the plain activation layer, both input and output vectors remain unencrypted, but they are encrypted in ciphertext activation layer. The approximate algorithms of both the ReLU and Sigmoid activation function with two degrees of approximation are described below in Algorithm 1 and 2.

Algorithm 1: ReLU Function

```
$\boldsymbol{Input}$: $in;\ n\_input;\ n\_output$
$\boldsymbol{Output}$: $out$
$scale = 10000$
$c_0 = 0$
$c_1 = 5000$
$c_2 = 55$
$in\_size \Leftarrow n\_input$
$\boldsymbol{for}\ j = 0,1,\dots\ in\_size\ do$
    $out[j] \Leftarrow in[j] * in[j] * c_2 + in[j] * c_1 + c_0$
end
```

Algorithm 2: Sigmoid Function

```
$\boldsymbol{Input}$: $in;\ n\_input;\ n\_output$
$\boldsymbol{Output}$: $out$
$scale = 10000$
$c_0 = 5000$
$c_1 = 5700$
$c_2 = -300$
$in\_size \Leftarrow n\_input$
$\boldsymbol{for}\ j = 0,1,\dots\ in\_size\ do$
    $out[j] \Leftarrow in[j] * in[j] * c_2 + in[j] * c_1 + c_0$
end
```

5.3.2 Convolution Layer Design

This layer gets a 3-D input vector, a 3-D weights vector and produces a 3-D vector as an output. The output vector has type long or ciphertext depends on whether an input to the layer is unencrypted or encrypted, respectively. In this layer, a sliding filter/kernel is utilized to compute the dot product between the weight vector and input vectors. The result of dot product is then elementwise added with the bias vector. The Algorithm 3 describes the steps involved in the convolutional layer. In plain convolution layer each vector (the input, weight, bias and the output) is unencrypted. The input/output vectors are encrypted in ciphertext convolution layer, but the weights and bias vectors are not, while all these vectors are not encrypted in plain convolutional layer. In SEAL, plaintexts and ciphertext can be added to and multiplied with one another.

Algorithm 3: Convolution Layer

```
$\boldsymbol{Input}$: $in$; $in\_height$; $in\_width$; $depth$; $kernel\_height$; $kernel\_width$; $n\_kernels$;
$weight$; $bias$; $scale$
$\boldsymbol{Output}$: $out$
$count \Leftarrow 0$
$out\_height \Leftarrow in\_height - (kernel\_height - 1)$
$out\_width \Leftarrow (in\_width - (kernel\_width - 1)$
$for\ k = 0,1,\dots n\_kernels\ do$
    $for\ y = 0,1,\dots\ out\_height\ do$
        $for\ x = 0,1,\dots\ out\_width\ do$
            $for\ c = 0,1,\dots\ depth\ do$
                $for\ ky = 0,1,\dots\ kerenel\_height\ do$
                    $for\ kx = 0,1,\dots\ kerenel\_width\ do$
                        $temp \Leftarrow in[y + ky][x + kx][c]$
                        $temp \Leftarrow temp * weight[ky][kx][c][k]$
                        $if\ count = 0\ then$
                            $out[y][x][k] \Leftarrow temp$
                            $count + +$
                        $else$
                            $out[y][x][k] += temp$
                            $count + +$
                        $if\ count = (kernel\_height * kernel\_width * depth)\ then$
                            $count = 0$
            $out[y][x][k] \Leftarrow b[k] * scale$
```

5.3.3 Pooling Layer Design

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 sum-pooling layer as shown in the Algorithm 4. The output of sum-pooling layer is simply the summation of values within the sliding window. This layer accepts a 3-D vector as input, and produces outputs of 3-D vector. The output vector is either a 3-D long vector or a 3-D ciphertext vector depends on if an input to the layer is unencrypted or encrypted, respectively. In the plain pooling layer, both input/output vectors are unencrypted, but they are encrypted in ciphertext pooling layer.

Algorithm 4: Pooling Layer

```
$\boldsymbol{Input}$: $in$; $in\_height$; $in\_width$; $depth$; $poolx$; $pooly$
$\boldsymbol{Output}$: $out$
$count \Leftarrow 0$
$out\_height \Leftarrow in\_height / pooly$
$out\_width \Leftarrow in\_width / poolx$
$for\ c = 0,1, \dots\ depth\ do$
    $for\ y = 0,1, \dots\ out\_height\ do$
        $for\ x = 0,1, \dots\ out\_width\ do$
            $for\ i = 0,1, \dots\ pooly\ do$
                $for\ j = 0,1, \dots\ poolx\ do$
                    $temp \Leftarrow in[y * pooly + i][x * poolx + j][c]$
                    $if\ count = 0\ then$
                        $out[y][x][k] \Leftarrow temp$
                        $count + +$
                    $else$
                        $out[y][x][k] += temp$
                        $count + +$
                    $if\ count = (pooly * poolx)\ then$
                        $count = 0$
```

5.3.4 Fully Connected Layer Design

This layer takes a 1-D weights vector, a 1-D input vector as input, and outputs 1-D vector. The output vector is either a 1-D long vector or a 1-D ciphertext vector depends on if an input to the layer is unencrypted or encrypted, respectively. This layer calculates the input vector's dot product with weight vector, then adds bias vector to outcome elementwise as shown in Algorithm 5. The weight vector and bias vector are not encrypted in fully connected ciphertext layer, but input/output vectors are encrypted, whereas all these vectors are not encrypted in plain fully connected layer.

Algorithm 5: Fully Connected Layer

```
$\textbf{Input}$: $in;\ n\_input;\ n\_output; weight;\ bias;\ scale$
$\textbf{Output}$: $out$
$in\_size \Leftarrow n\_input$
$out\_size \Leftarrow n\_output$
$for\ i = 0,1, \dots\ out\_size\ do$
    $for\ j = 0,1, \dots\ in\_size\ do$
        $temp \Leftarrow in[j]$
        $temp \Leftarrow temp * weight[j][i]$
        $if\ j = 0\ then$
            $out[i] \Leftarrow temp$
        $else$
            $out[y][x][k] += temp$
$out[i] \Leftarrow b[i] * scale$
```

Chapter 6

Experimental Results and Evaluation

The major motivation for developing privacy-preserving CNN classification model is to ensure information secrecy for all involved parties. There are instances that demand privacy protection measures, even if they are not necessary in most cases. For instance, while working with the medical data, confidentiality of a patient personal information is of utmost importance. In this case, a privacy-preserving CNN classification model may be utilized to let patients get a diagnosis by transmitting his personal information, with patient being the only one who can access the information and the diagnostic. In addition, the hospital may use its classifier on the encrypted data while still keeping its model hidden from the patients.

In this chapter, first we approximate the activation functions on different scales and degrees and present the graphical representation of original and approximated activation functions. Next, we analyze the effect of variation of HE parameters on the time and accuracy of proposed CNN classification model. At the end we discuss the fast configurations required for the proposed model in term of time and accuracy.

6.1 Polynomial Approximation Techniques

There are numerous methods available for approximating a continuous function, but we are only concerned with polynomial approximation in this work. For polynomial approximation, a number of techniques have been put forth in the literature, such as Taylor series and Chebyshev polynomials [70][71]. We examine the following approaches to approximate the activation functions:

- Numerical approximation method
- Taylor series method
- Chebyshev approximation method

We employ each of these techniques individually to analyze the polynomial approximation of activation functions and also the merit and demerits of these methods are discussed below.

Numerical Analysis: In this technique, the set of points is produced from the activation function and feed this set into approximation function, along with a constant degree for activation function. We tested polynomials with degree ranging 3-10 and concluded that the accuracy declines significantly for lower degree polynomials. To get optimal accuracy, we must raise the degree, which is inefficient when dealing with encrypted data. Our analysis revealed that this technique is not an effective way for approximating the activation function.

Taylor Series: Taylor series, a prominent approach for estimating functions, is used in this method. We approximated the activation function using polynomials of varying degrees and trained the model with polynomials of given degrees. This approach is ineffective due to two key problems. The first problem is that, despite being lower than 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 approximation of given functions in a point nearby space. Approximation error is significantly larger for the points outside of the input interval than for those within of it. For instance, this approach is unable to cover the $[0, 255]$ range of integer values for pixels in the MNIST dataset. Chebyshev polynomials [71] may be used to estimate the activation function across a wide range, which allows us to avoid requiring further layers, as explained below.

Chybeshev approximation Method: The use of Chebyshev polynomials is not as widespread as earlier techniques. But these are more appropriate for our problem due to a certain feature. With this approach, we estimate a function throughout an interval rather than just a tiny region around a point. We increase the interval to be able to cover integers since HE systems are over integers with message space $\mathbb{Z}$. The Chybeshev polynomial is given as:

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

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 [72]. 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. In order to identify a mathematical function that minimizes the maximum error, the minimax approximation is applied. As an example, for a function f defined over the interval $[a, b]$, the minimax approximation finds a polynomial $p(x)$ that minimizes max $\max_{a \le x \le b} |f(x) - p(x)|$.

In order to approximate a continuous function f, defined over $[a, b]$, we need to describe f as a sequence of Chebyshev polynomials at $[-1,1]$. More precisely, f is expressed as:

$$f(x) = \sum_{k=0}^{n} c_k T_k(x) \qquad\qquad x \in [-1, 1]$$

where c_k is the Chebyshev coefficient and $T_k(x)$ can be calculated from above mentioned equation. The polynomial's coefficients are then calculated, and it is eventually expressed in the original interval $[a, b]$.

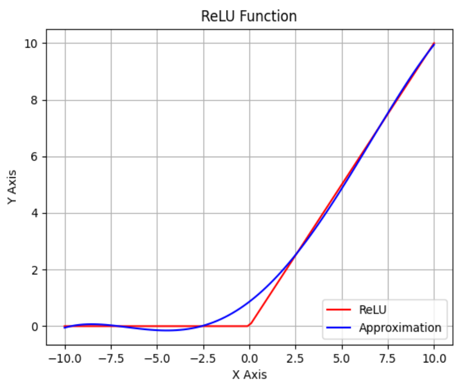
6.2 Activation Functions Approximation

We employ Chebyshev polynomial approximate method to approximate the ReLU and Sigmoid activation functions in our proposed model, in which the inputs are HE encrypted images. Table 5 depicts the polynomial approximation of the ReLU activation function with degree 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 a number of experiments with various intervals and degrees.

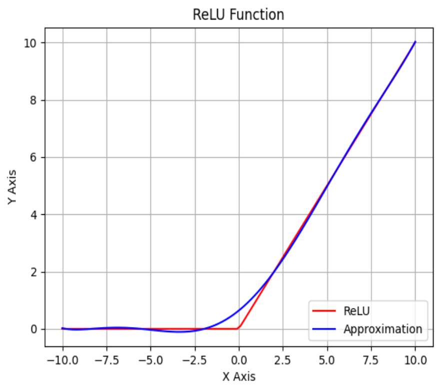
Table 5 demonstrates that the activation functions are more precisely approximated when higher degree polynomials are used in short intervals. For instance, compared to the other polynomials, the polynomial with degree 7 and interval $[10, 10]$ is more accurate in its approximation of the ReLU function. The same holds true for the Sigmoid activation function, where a high degree 7 and short interval $[10, 10]$ provide a better approximation, as shown in Table 5. However, using higher degree polynomials imposes substantial computation overhead, and short intervals limit the approximation function's applicability.

Table 5. *Approx. of ReLU Function on Two Intervals Using Degree 5 & 7*

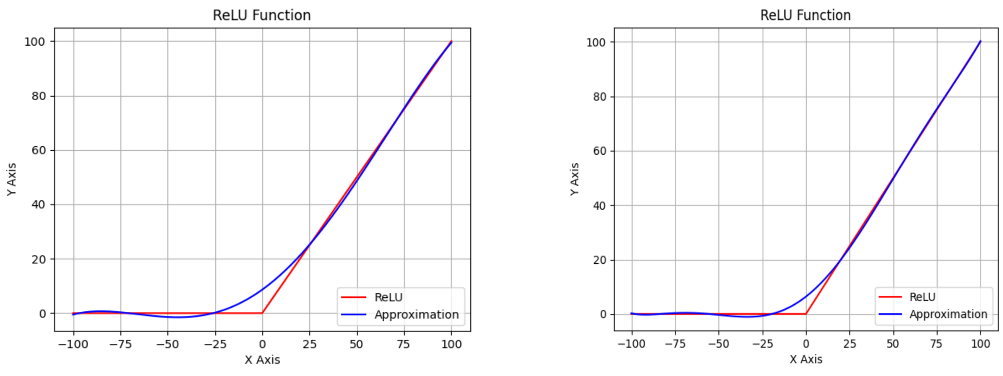
Degree	**Intervals**	**Polynomial Approximation of Activation Function**	**ReLU Functions**
5	$[-10, 10]$	$(2.368475785867e^{-19}) \times x^5 - (0.000252624921308674) \times x^4 - (2.90138283768708e^{-17}) \times x^3 + (0.0660873211772537) \times x^2 + (0.500000000000001) \times x + (0.862730150341736)$	Fig 15(a)
7	$[-10, 10]$	$(-8.88178419700125e^{-21}) \times x^7 + (3.66197231323541e^{-6}) \times x^6 + (1.33226762955019e^{-18}) \times x^5 - (0.000847927183186682) \times x^4 - (5.24025267623074e^{-17}) \times x^3 + (0.0920352084972136) \times x^2 + (0.500000000000001) \times x + (0.637244473880199)$	Fig 15(b)
5	$[-100, 100]$	$(2.27373675443232e^{-23}) \times x^5 - (2.52624921308674e^{-7}) \times x^4 - (2.70006239588838e^{-19}) \times x^3 + (0.00660873211772537) \times x^2 + (0.500000000000001) \times x + (8.62730150341737)$	Fig 15(c)
7	$[-100, 100]$	$(-6.82121026329696e^{-27}) \times x^7 + (3.6619723132354e^{-11}) \times x^6 + (1.03739239420975e^{-22}) \times x^5 - (8.47927183186682e^{-7}) \times x^4 - (4.2277292777726e^{-19}) \times x^3 + (0.00920352084972135) \times x^2 + (0.5) \times x + (6.37244473880199)$	Fig 15(d)



(a)



(b)



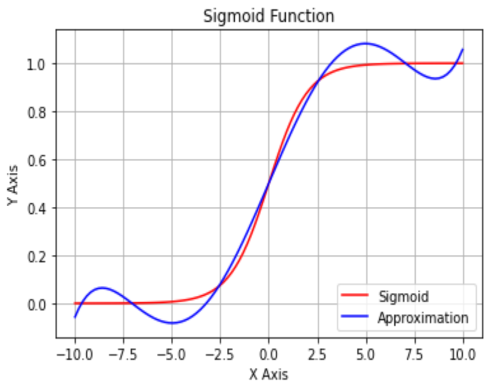
(c) *(d)*

***Figure 15**. Approximation of ReLU Functions*

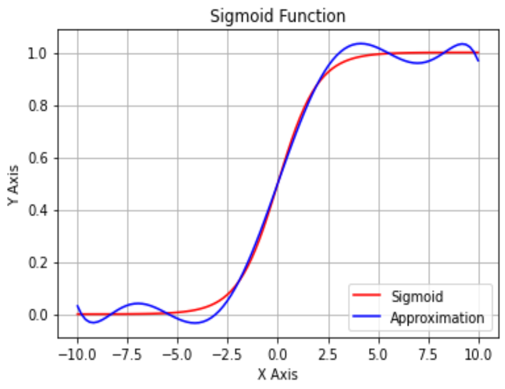
Table 6 depicts the polynomial approximation of the Sigmoid activation function.

***Table 6**. Approx. of Sigmoid Function on Two Intervals Using Degree 5 & 7*

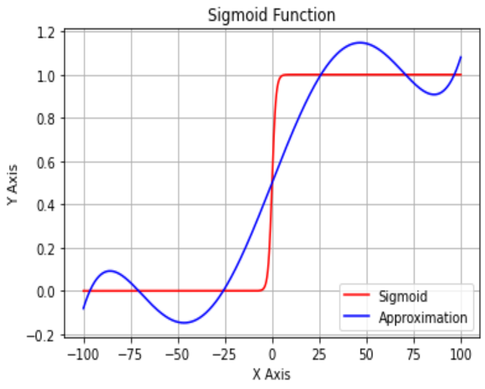
Degree	**Intervals**	**Polynomial Approximation of Activation Function**	**Sigmoid Functions**
5	$[-10, 10]$	$(2.04674243304457e^{-5}) \times x^5 + (2.46800554530117e^{-20}) \times x^4 - (3.36794817460072\, e^{-3}) \times x^3 - (1.8874604570257e^{-18}) \times x^2 + (0.18781951515784) \times x + 0.5$	Fig 16(a)
7	$[-10, 10]$	$(-4.3491363562486e^{-7}) \times x^7 - (2.72617215260618e^{-21}) \times x^6 + (9.1841913854224e^{-5}) \times x^5 + (2.97822205265474e^{-19}) \times x^4 - (0.00652613009718176) \times x^3 - (4.60152342661793e^{-18}) \times x^2 + (0.216030242319584) \times x + (0.5)$	Fig 16(b)
5	$[-100, 100]$	$(2.76073648103432e^{-10}) \times x^5 - (5.79639277612257e^{-24}) \times x^4 - (4.39372964286412e^{-6}) \times x^3 + (8.80835665108732e^{-20}) \times x^2 + (0.0221378748236003) \times x + (0.5)$	Fig 16(c)
7	$[-100, 100]$	$(-8.15672915997047e^{-14}) \times x^7 - (2.69310298657131e^{-27}) \times x^6 + (1.66796555552873e^{-9}) \times x^5 + (2.88883265216498e^{-23}) \times x^4 - (1.10438386410423e^{-5}) \times x^3 - (2.84535249558831e^{-20}) \times x^2 + (0.0295953437969288) \times x + (0.5)$	Fig 16(d)



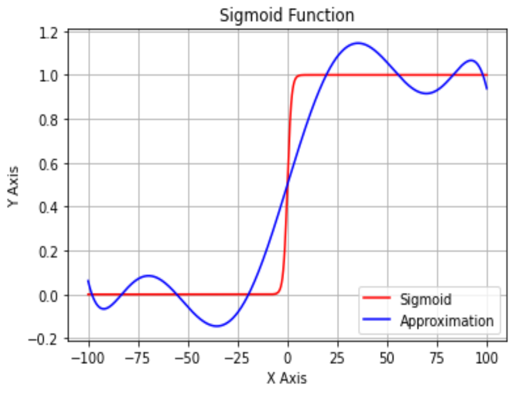
(a)



(b)



(c)



(d)

Figure 16. *Polynomial Approximation of Sigmoid Functions*

6.3 CNN Model Accuracy with Polynomial Activation Function

We conduct our experiments using the MNIST [6] dataset and the CNN model that is described below in Fig 18 to assess performance of various approximation methods. For a comparison, we first train the given model using original activation functions and find the accuracy against each activation function as shown in Table 7. Then, we use the approximation polynomial of degree two for the given activation functions in the given model and calculate the accuracies given in the Table 7.

Table 7. CNN accuracies based on original and approximated activation functions

Activation Function	Original Model	Approximated Model
Sigmoid	98.82 %	98.25%
ReLU	99.15%	98.75%

6.4 Privacy Preserved Classification Model

In this section we discussed the different part involved in the proposed privacy preserved classification model. It includes the dataset we used for the training purpose and details about the structure of proposed CNN model as well as the input and output sizes of each layer present in it. It also contains the model training and testing part along with the tuning parameters.

6.4.1 Dataset

The MNIST data set is used to train and test the privacy-preserving CNN. This dataset is chosen specifically because it is widely used in the field of deep learning. This enable a comparison of accuracy with prior research. There are 60,000 total images in this dataset, from which 50,000 images are choosen for training while remaining 10,000 images are selected for testing. The MNIST dataset contains images of $28x28$ pixel arrays, every pixel is made up of a positive integer ranging from 0-255. Fig 17 displays a sample of pictures from the MNIST dataset.

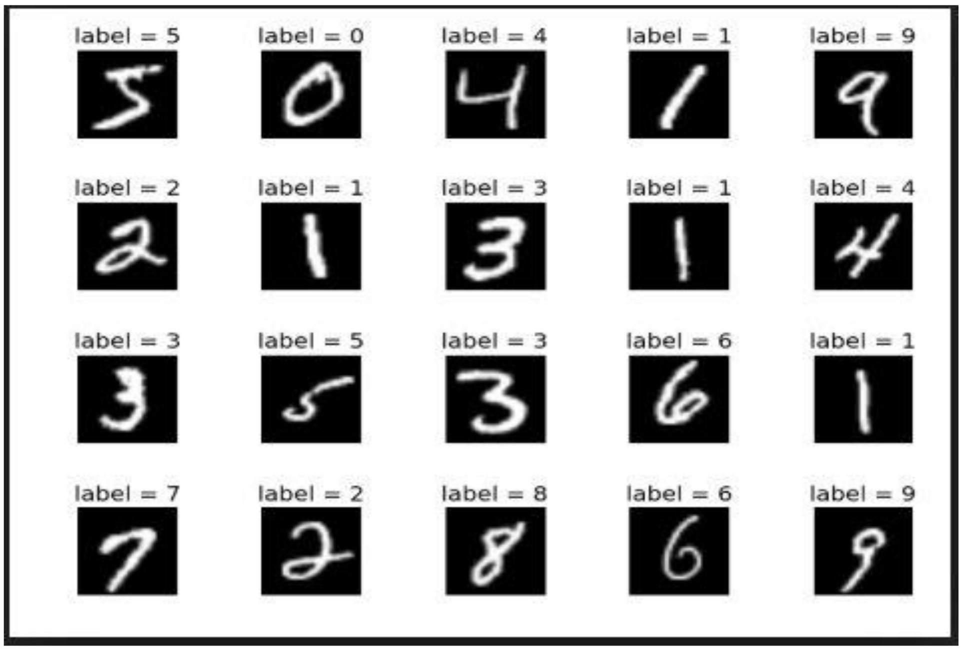


Figure 17. MNIST Dataset

6.4.2 CNN Network

The CNN network which is used to train and categories MNIST dataset is shown in Fig 18. A summary of this network is provided below:

1. 1st Conv-Layer: It takes an image of dimension $28 \times 28 \times 1$ as an input. The layer contains 4 kernels of dimension 5×5, with stride of $(1,1)$.
2. Activation-Layer: It performs the ReLU function at every input node.
3. 1st Pool-Layer: It takes an input of dimension $24 \times 24 \times 4$ and have stride of size (2,2). Its output is $12 \times 12 \times 4$.
4. 2nd Conv-Layer: It has input of dimension 12×12×4. This layer consists of 12 kernels of dimension 5×5, and stride of $(1,1)$. The outcome of this layer is $8 \times 8 \times 12$.
5. 2nd Pool-Layer: It takes an input of dimension $8 \times 8 \times 12$ and have stride of size (2,2). Its output is $4 \times 4 \times 12$.
6. Flatten-Layer: The input of this layer is $4 \times 4 \times 12$ and returns the output of 192.
7. Fully Connected-Layer: It combines incoming 192 notes to the 10 output nodes.

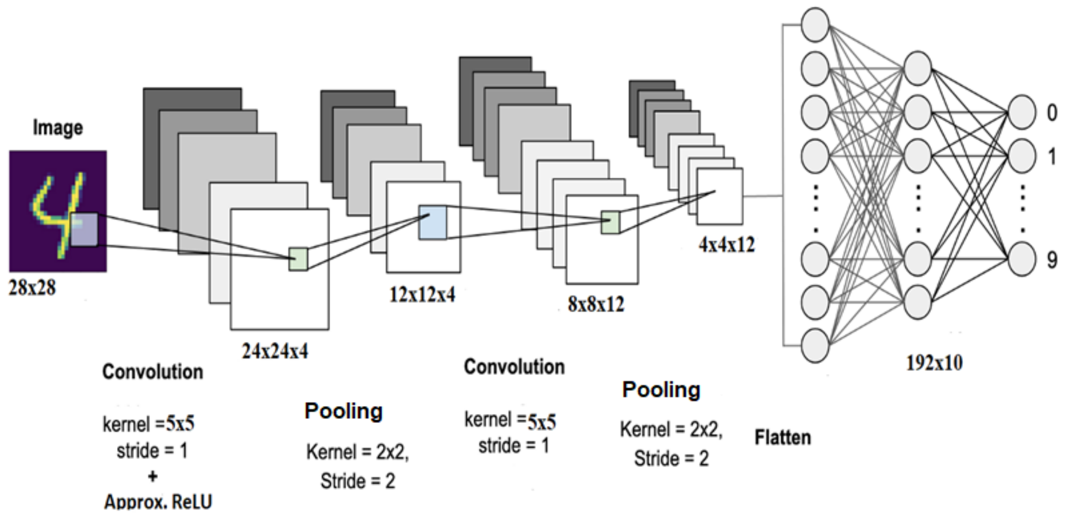


Figure 18. *Proposed Convolutional Neural Network*

6.4.3 Model Training

The original ReLU function is replaced by Chybeshev approximated ReLU function during learning phase. The CNN model is trained with the PyTorch framework using PyTorch library with MNIST dataset. 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 hyper parameter adjustment.

6.4.4 Model Testing

The ReLU activation function is changed by degree two polynomial approximation function during the classification phase, while the pooling layer is changed to the sum-pooling layer. 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 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 aren't classified in batches since working with encrypted data requires a considerable amount of processing power and memory. Instead, the privacy-preserving classier processed each image separately.

6.5 Result Profiling

Accuracy and Timing are two of the most crucial aspects to examine when evaluating the practicality of any cryptosystem: does this accurately categories the image as well as how much time does it take to classify the image? 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 the capabilities of HE. Timing is determined by counting the seconds it took to perform each layer and to encrypt/decrypt the image. In order 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 6,000 images.

6.5.1 Timing

It is discovered during the initial testing phases that our privacy preserved CNN model takes 215.08 seconds to classify an encrypted image. We calculate the time it took for encryption/decryption of an image as well as time of each layer in order to determine where it might be spending the most of its time during the classification phase. Table 8 and 9 show the time needed to read the image, encrypt it, execution time of each layer, and the decryption of final result. The security parameter is set to 128 bits for all timing values indicated below. The execution time of each layer of our CNN model is shown in the Table 8.

Table 8. *Running Time 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

The convolution layer requires the most computation time, as shown in Table 8. The activation layer is the 2nd most expensive layer in term of time, then comes fully connected layer, and lastly the pooling layer.

Table 9 shows the time the model takes to read, encrypt and decrypt the image.

Table 9. *Image Encryption/Decryption Time*

Operation	**Time (s)**
Read Image	0.0005
Encrypt Image	3.28
Decrypt	0.02

According to the timing data in Table 9, reading the image takes only 0.0005 seconds, which is a very little amount of time. Encryption takes 3.28 seconds, hence it takes around 3.28/(28*28*1) = 0.0128125 seconds to encrypt one pixel. Decryption takes 0.02 seconds, hence it takes around 0.2/(10) = 0.02 seconds to decrypt one value.

It is evident from the preliminary time data shown in Table 8 that the convolution layer and ReLU activation layers are the main sources of the computational bottleneck.

6.5.2 Security Parameter (k/λ) Variation

In the prospective of model security, HE has a few factors that are essential to consider while performing the CNN classification. One of these parameters is k/λ, which is called the security parameter. We use the default value of $k = 128$ for experiments. The security parameters are varied in this section to observe the threshold (Time & Accuracy) for calculation as well as the overall security. The timing values are calculated by using BFV scheme for polynomial modulus degree n = 8192, for each three security parameters of 128, 192 and 256 as recommended in [73].

For SEAL, setting $k = 128$ is comparable to AES 128-bit security, setting $k = 192$ is equivalent to AES 192-bit security, and setting $k = 256$ is similar to AES 256-bit security. Therefore, in addition to the default k = 128, these are the two other security settings assessed.

Fig 19 depicts the time takes by each CNN layer to run based-on the variation security parameters. According to the stats in Fig 19, the time required to evaluate each layer increases as the security parameter increases. When security parameter is increased from 128-bits to 256-bits, the calculation time for the layers (Conv1/Conv2/FC1) that take the longest to calculate increases by 1.8 times.

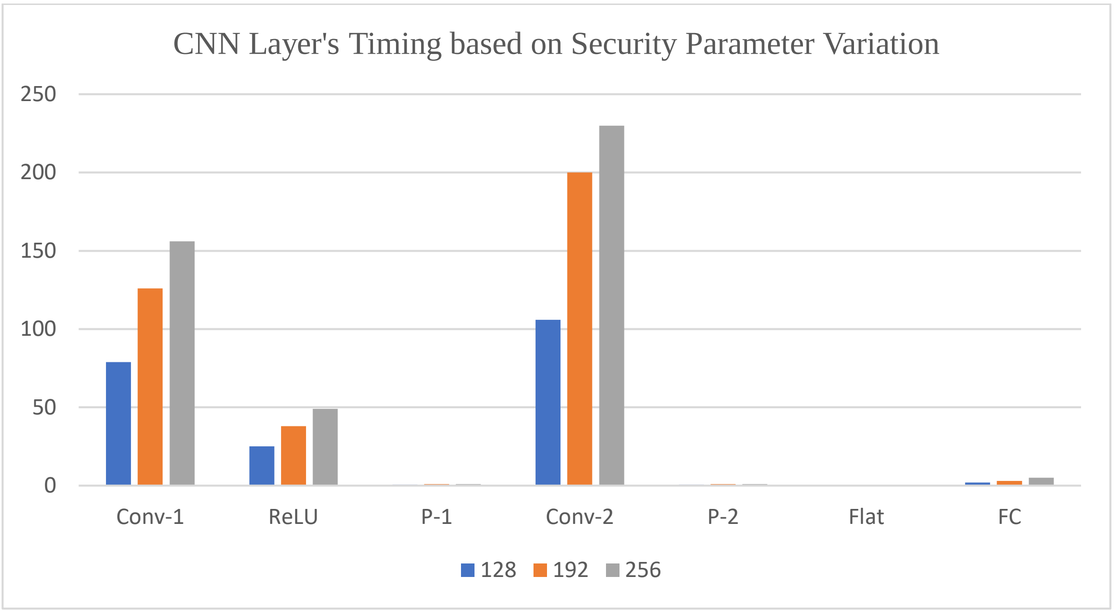


Figure 19. *Execution Time of different layers based on Security Parameters Variation*

The total time needed to run the network with regard to changing security parameters is shown in Fig 20 below.

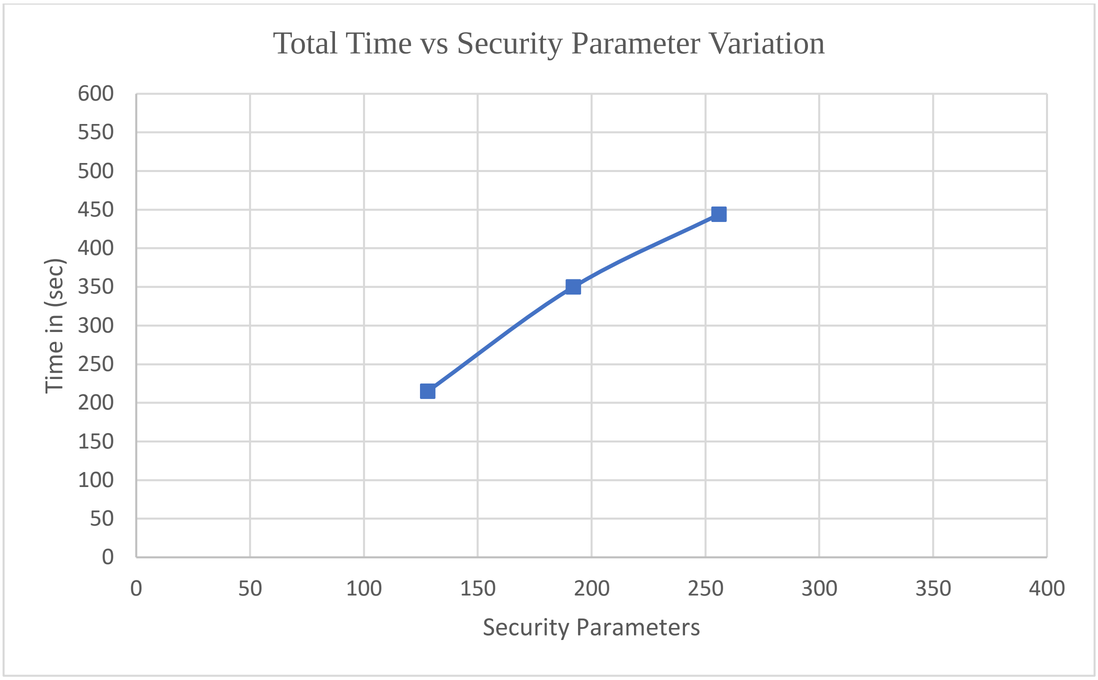


Figure 20. *Total Execution Time of model based on Security Parameter Variation*

From Fig 20 it is evident that the computation time increases as the security parameter size increases. The graph suggests that there is a linear relationship with a gradual slope between the value of the security parameter and the overall classification time. The overall time to classify an encrypted image with a security parameter of 128-bit is 215.08 seconds, with a 192-bit security parameter is 302.26 seconds, and 444.84 seconds with a 256-bit security parameter. It is significant to notice that the encrypted image could not be fully classified using the security parameter of 256 because the noise increases too high and there are insufficient levels to support the given security parameter. It takes about 230 seconds longer to use 256-bit security parameter than 128-bit security parameter. This raises the issue of time at the expense of security: under what circumstances would someone be ready to wait even longer in order to get higher security level?

Based on the sizes of the security parameter, Table 10 indicates time taken by the model to read, encrypt, and decrypt the given image. It also determines whether or not the image is properly classified.

Table 10. *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

According to the timing data in Table 10 security parameter change also has an influence on the encryption and decryption time of image. Time required for encryption of image increases with the increase in the security parameter. In the large context of scheme things, 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.

6.5.3 Other parameters (q & n) variations

As discussed in Chapter 4 the ciphertext/coefficient modulus q size has a direct relation with the noise budget. As complex computations require more noise budget, so a larger coefficient modulus is required. The coefficient modulus in SEAL is a positive composite number that is the sum of several primes with a maximum bit size of 60 bits. These primes are also called the number of levels of modulus chain. These levels (primes) in coefficient modulus are changed after each multiplication operation. This implies that the evaluation functions have a significant impact on the level values. So, a complex evaluation functions require larger values of the coefficient modulus. However, a higher coefficient modulus q also reduces the level of security of the schemes. Therefore, we have to increase the value of polynomial modulus n value at the same time to meet the required security level. Conversely, the size of polynomial modulus n and number of prime elements in coefficient modulus q have the greatest influence on performance. The relationship between the polynomial modulus degree n

and the corresponding upper bound of coefficient modulus q (Number of levels) for different security level is provided in [73]. According to this recommendation, as the polynomial modulus degree values increase the corresponding ciphertext modulus values change and so the number of levels.

In this section, both the values of coefficient modulus q and polynomial modulus n are varied to observed the change in threshold values and overall time. All timing values are measured by using BFV scheme for different values of n (1024, 2048, 4096, 8192, 16384) with standard security parameters of 128-bit.

Fig 21 illustrates the total execution 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 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).

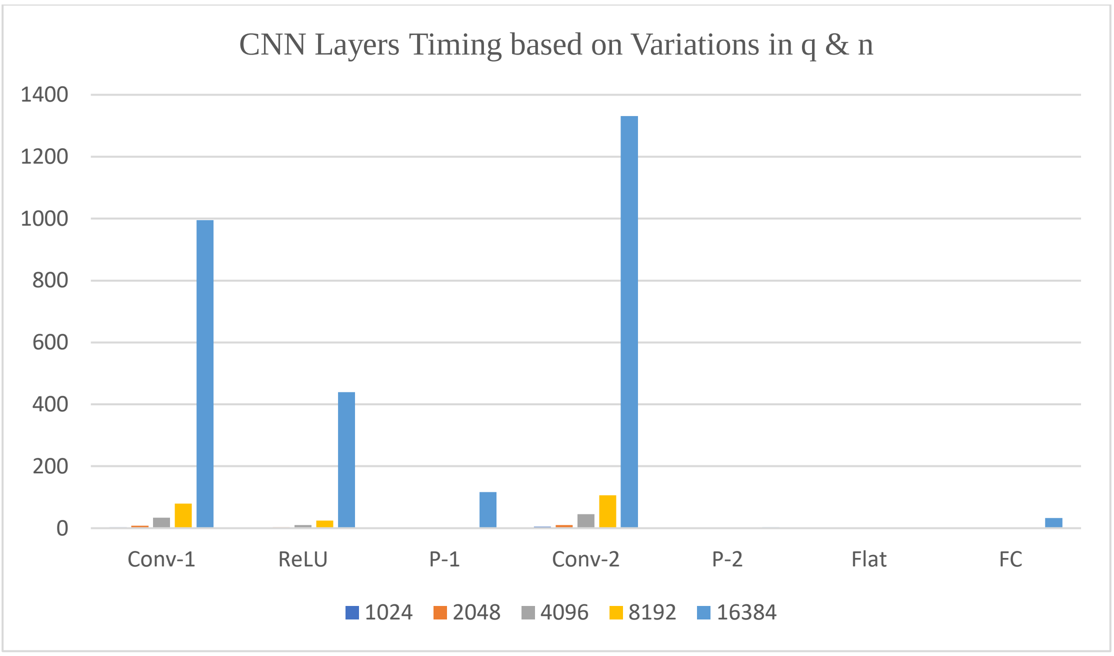


Figure 21. *Execution Time of different layers based on Polynomial Degree Variation*

Fig 22 clearly demonstrates the total execution time of network based on variation in polynomial modulus degree n and the corresponding number of levels in coefficient modulus q. The findings in Figure 22 illustrates that the computation time increases with the increase in polynomial modulus degree n and the corresponding number of levels in coefficient modulus q. The polynomial modulus degree n and time of classification seems to be correlated linearly with an average slope, as shown in the graph. The same is true for varying the number of levels in coefficient modulus q. It is significant to notice that if there are polynomial degree n is less than 4096, the noise increase would have been not enough for the network to successfully categorize the encrypted image. Additionally, Figure 22 slope is substantially steeper than Figure 20 slope. It makes more sense to reduce the number of layers over the security parameter as much as feasible in order to reduce computation time.

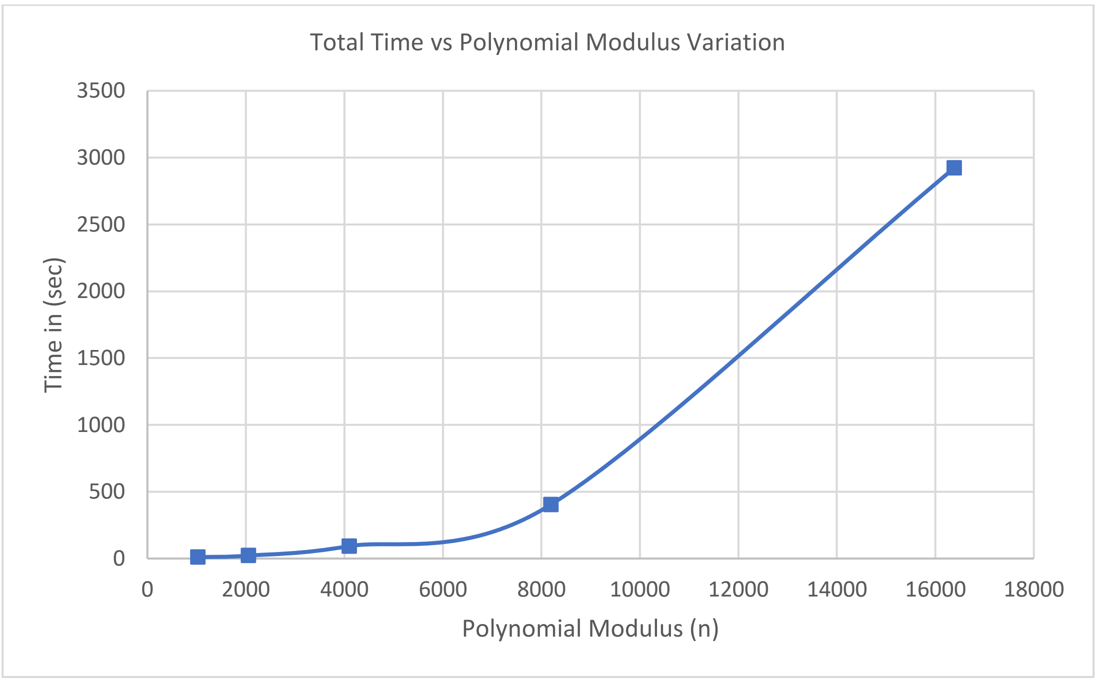


Figure 22. *Total Execution Time of model based on Polynomial Modulus Variation*

Table 11 shows how long it takes 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 data in Table 8 clearly shows that polynomial modulus degree n and coefficient modulus q do have an effect the encryption and decryption time of image. This time increases as the values of polynomial modulus degree n and coefficient modulus q increase. In the broad scheme of things, this small difference in time is insignificant.

Table 11. *Encryption/Decryption Time*

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

Accuracy is also affected by varying the size of polynomial modulus degree n and coefficient modulus q. A certain minimum number of levels corresponding the polynomial modulus degree n is undoubtedly required to effectively categorize the given encrypted image and control the noise budget. There is not simple formula for calculating the required number of levels in coefficient modulus q. But in general, the number of levels must be equivalent to the number of multiplications in the evaluation function. The network put to the test in this experiment includes a degree two polynomial computation and three dot products. Because of this, early tests were conducted with different levels set. The minimal number of levels was found by guessing and checking when results displayed an error message.

Thus, based on experiments, it is recommend using as few factors in the coefficient modulus is possible for good performance.

6.6 Best Configuration

By understanding all the above-mentioned results and limitations of HE, final test for the fast configuration is carried out. Best configuration means to select 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 security parameter set to 128-bit.

Table 12. *Timing of each layer based on Best Configuration*

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

Table 12 represents the all-time best value of each layer execution time based on the best parameter selection, while still maintaining the correct prediction.

Chapter 7

Conclusions and Future Work

We conclude our thesis by providing a concrete conclusion and future work in two sections. The first section contains the brief overview of the thesis. The second section includes the future work in the context of privacy-preserving classification through CNN using HE.

7.1 Conclusion

In this thesis, 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, specifically for employing CNN for classification in an outsourced setting, were investigated. To keep these limitations in mind, we modified the different layer of CNN to make them compatible with HE supported operations. In particularly, we approximated the non-linear activation functions like Sigmoid and ReLU into functions which only includes additions and multiplications terms. We approximated these function on different degree and scales to examine the impact of these variations on accuracy of proposed classification model. At the end, we calculated the time of each layer and the overall time as well as the accuracy of proposed model by varying the HE parameters. Overall, this work served as an effective demonstration of concept for the classification of encrypted images.

7.2 Future Work

The current study has opened several pathways for future research. Some of them are summarized below.

In our current research, we have been working only on feed-forward neural networks. The most prominent example is the CNN. However, there are numerous other neural network architectures, like recurring neural networks, are available which can be explored for evaluation using homomorphic encryption.

Our proposed model only included two entities the client and the server i.e., we only consider one to one client/server architecture. This work would be expanded in the multi-party setting using multi-key homeomorphic encryption schemes.

In our suggested study, we took into account a semi-honest threat model architecture. The participants closely adhere to the protocol's rules in the semi-honest environment, but they are still interested in learning about other inputs from the interactions. So, to move beyond this threat model, like malicious threat model, would be an interesting work to do.

References

[1] E. Kreke, “From Directive 95/46/EC to the General Data Protection Regulation: Addressing the potential harm to data subjects’ rights arising from personal data collection and data analytics,” 2018.

[2] Q. Zhang, L. T. Yang, and Z. Chen, “Privacy preserving deep computation model on cloud for big data feature learning,” *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1351–1362, 2015.

[3] 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 *Proceedings of The 33rd International Conference on Machine Learning*, New York, New York, USA, vol. 48, pp. 201–210.

[4] H. Chabanne, A. De Wargny, J. Milgram, C. Morel, and E. Prouff, “Privacy-preserving classification on deep neural network,” *Cryptology ePrint Archive*, 2017.

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

[6] Y. LeCun, “The MNIST database of handwritten digits,” *http://yann. lecun. com/exdb/mnist/*, 1998.

[7] K. Laine, “Simple encrypted arithmetic library 2.3. 1, Microsoft Research, 2017.”

[8] Z. Brakerski, “Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP,” in *Advances in cryptology – CRYPTO 2012*, vol. 7417, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 868–886.

[9] J. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” *Cryptology ePrint Archive*, 2012.

[10] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Advances in cryptology – ASIACRYPT 2017*, vol. 10624, T. Takagi and T. Peyrin, Eds. Cham: Springer International Publishing, 2017, pp. 409–437.

[11] H. Chen, K. Laine, and R. Player, “Simple Encrypted Arithmetic Library - SEAL v2.1,” in *Financial cryptography and data security*, vol. 10323, M. Brenner, K. Rohloff, J. Bonneau, A. Miller, P. Y. A. Ryan, V. Teague, A. Bracciali, M. Sala, F. Pintore, and M. Jakobsson, Eds. Cham: Springer International Publishing, 2017, pp. 3–18.

[12] A. Ibarrondo and A. Viand, “Pyfhel: Python for homomorphic encryption libraries,” in *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2021, pp. 11–16.

[13] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, doi: 10.1145/359340.359342.

[14] J. Katz and Y. Lindell, *Introduction to modern cryptography*. Chapman and Hall/CRC, 2020.

[15] C. Fontaine and F. Galand, “A survey of homomorphic encryption for nonspecialists,” *EURASIP J. on Info. Security*, vol. 2007, pp. 1–10, 2007, doi: 10.1155/2007/13801.

[16] T. Elgamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” *IEEE Trans. Inform. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985, doi: 10.1109/TIT.1985.1057074.

[17] J. Benaloh, “Dense probabilistic encryption,” in *Proceedings of the workshop on selected areas of cryptography*, 1994, pp. 120–128.

[18] P. Paillier, “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes,” in *Advances in cryptology — EUROCRYPT ’99*, J. Stern, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238.

[19] S. Goldwasser and S. Micali, “Probabilistic encryption & how to play mental poker keeping secret all partial information,” in *Proceedings of the fourteenth annual ACM symposium on Theory of computing - STOC ’82*, New York, New York, USA, 1982, pp. 365–377, doi: 10.1145/800070.802212.

[20] I. Damgård and M. Jurik, “A generalisation, a simplification and some applications of Paillier’s probabilistic public-key system,” in *International workshop on public key cryptography*, 2001, pp. 119–136.

[21] D. Boneh, E.-J. Goh, and K. Nissim, “Evaluating 2-DNF formulas on ciphertexts,” in *Theory of cryptography conference*, 2005, pp. 325–341.

[22] T. S. Fun and A. Samsudin, “A survey of homomorphic encryption for outsourced big data computation,” *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 10, no. 8, pp. 3826–3851, 2016.

[23] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC ’09*, New York, New York, USA, 2009, p. 169, doi: 10.1145/1536414.1536440.

[24] N. P. Smart and F. Vercauteren, “Fully homomorphic encryption with relatively small key and ciphertext sizes,” in *International Workshop on Public Key Cryptography*, 2010, pp. 420–443.

[25] A. López-Alt, E. Tromer, and V. Vaikuntanathan, “On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption,” in *Proceedings of the 44th symposium on Theory of Computing - STOC ’12*, New York, New York, USA, 2012, p. 1219, doi: 10.1145/2213977.2214086.

[26] Z. Brakerski and V. Vaikuntanathan, “Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages,” in *Advances in cryptology – CRYPTO 2011*, vol. 6841, P. Rogaway, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 505–524.

[27] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, “Fully Homomorphic Encryption over the Integers,” in *Advances in Cryptology – EUROCRYPT 2010*, vol. 6110, H. Gilbert, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 24–43.

[28] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, Jan. 1999, doi: 10.1137/S0036144598347011.

[29] W. Diffie and M. E. Hellman, “New directions in cryptography,” in *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, 2022, pp. 365–390.

[30] N. Koblitz, “Elliptic curve cryptosystems,” *Math. Comput.*, vol. 48, no. 177, pp. 203–203, Jan. 1987, doi: 10.1090/S0025-5718-1987-0866109-5.

[31] V. S. Miller, “Use of elliptic curves in cryptography,” in *Advances in cryptology — CRYPTO ’85 proceedings*, H. C. Williams, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 417–426.

[32] R. J. McEliece, “A public-key cryptosystem based on algebraic,” *Coding Thv*, vol. 4244, pp. 114–116, 1978.

[33] J. Hoffstein, J. Pipher, and J. H. Silverman, “NTRU: A ring-based public key cryptosystem,” in *International algorithmic number theory symposium*, 1998, pp. 267–288.

[34] M. Ajtai, “Generating hard instances of lattice problems,” in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 99–108.

[35] M. Ajtai, “The shortest vector problem in L2 is NP-hard for randomized reductions,” in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 10–19.

[36] O. Regev, “Lattice-based cryptography,” in *Annual International Cryptology Conference*, 2006, pp. 131–141.

[37] C. Peikert, “A decade of lattice cryptography,” *FNT in Theoretical Computer Science*, vol. 10, no. 4, pp. 283–424, 2016, doi: 10.1561/0400000074.

[38] S. Garg, C. Gentry, and S. Halevi, “Candidate Multilinear Maps from Ideal Lattices,” in *Advances in cryptology – EUROCRYPT 2013*, vol. 7881, T. Johansson and P. Q. Nguyen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–17.

[39] D. Micciancio and O. Regev, “Lattice-based cryptography,” in *Post-quantum cryptography*, Springer, 2009, pp. 147–191.

[40] S. D. Galbraith, *Mathematics of public key cryptography*. Cambridge University Press, 2012.

[41] D. Micciancio and S. Goldwasser, *Complexity of lattice problems: a cryptographic perspective*, vol. 671. Springer Science & Business Media, 2002.

[42] D. Micciancio, “The hardness of the closest vector problem with preprocessing,” *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 1212–1215, 2001.

[43] D. Micciancio, “The shortest vector in a lattice is hard to approximate to within some constant,” *SIAM journal on Computing*, vol. 30, no. 6, pp. 2008–2035, 2001.

[44] O. Regev, “On lattices, learning with errors, random linear codes, and cryptography,” *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[45] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Annual international conference on the theory and applications of cryptographic techniques*, 2010, pp. 1–23.

[46] "An Introduction To Mathematics Behind Neural Networks | by Gautham S | Analytics Vidhya | Medium." [Online]. Available: https://medium.com/analytics-vidhya/an-introduction-to-mathematics-behind-neural-networks-135df0b85fa1. [Accessed: 16-Nov-2022]

[47] "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science." [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 16-Nov-2022]

[48] "Simple Introduction to Convolutional Neural Networks | by Matthew Stewart | Towards Data Science." [Online]. Available: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac. [Accessed: 16-Nov-2022]

[49] H. Yingge, I. Ali, and K.-Y. Lee, "Deep Neural Networks on Chip - A Survey," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2020, pp. 589–592, doi: 10.1109/BigComp48618.2020.00016.

[50] "Affine Layer Definition | DeepAI." [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/affine-layer. [Accessed: 16-Nov-2022]

[51] V. Costan and S. Devadas, "Intel SGX explained," *Cryptology ePrint Archive*, 2016.

[52] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on intel SGX," in *Proceedings of the 10th European Workshop on Systems Security - EuroSec'17*, New York, New York, USA, 2017, pp. 1–6, doi: 10.1145/3065913.3065915.

[53] S. Samet and A. Miri, "Privacy-preserving classification and clustering using secure multi-party computation," in *Proceeding of the International Conference on Relations, Orders and Graphs: Interaction with Computer Science (ROGICS)*, 2008, pp. 482–491.

[54] H. C. A. van Tilborg and S. Jajodia, Eds., "Secure Multiparty Computation," in *Encyclopedia of cryptography and security*, Boston, MA: Springer US, 2011, pp. 1121–1121.

[55] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: machine learning on encrypted data," in *Information security and cryptology – ICISC 2012*, vol. 7839, T. Kwon, M.-K. Lee, and D. Kwon, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–21.

[56] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *Cryptology ePrint Archive*, 2014.

[57] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-nets: Neural networks over encrypted data," *arXiv preprint arXiv:1412.6181*, 2014.

[58] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1209–1222.

[59] Y. Aono, T. Hayashi, L. Wang, S. Moriai, and Others, “Privacy-preserving deep learning via additively homomorphic encryption,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[60] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015, pp. 909–910, doi: 10.1109/ALLERTON.2015.7447103.

[61] W. Liu, F. Pan, X. A. Wang, Y. Cao, and D. Tang, “Privacy-preserving all convolutional net based on homomorphic encryption,” in *International Conference on Network-Based Information Systems*, 2018, pp. 752–762.

[62] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, “$\{$GAZELLE$\}$: A low latency framework for secure neural network inference,” in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1651–1669.

[63] J. Liu, M. Juuti, Y. Lu, and N. Asokan, “Oblivious neural network predictions via minionn transformations,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS ’17*, New York, New York, USA, 2017, pp. 619–631, doi: 10.1145/3133956.3134056.

[64] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade, “TAPAS: Tricks to accelerate (encrypted) prediction as a service,” in *International Conference on Machine Learning*, 2018, pp. 4490–4499.

[65] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, “Secure logistic regression based on homomorphic encryption: design and evaluation.,” *JMIR Med. Inform.*, vol. 6, no. 2, p. e19, Apr. 2018, doi: 10.2196/medinform.8805.

[66] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, “Logistic regression model training based on the approximate homomorphic encryption.,” *BMC Med. Genomics*, vol. 11, no. Suppl 4, p. 83, Oct. 2018, doi: 10.1186/s12920-018-0401-7.

[67] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, “Fast homomorphic evaluation of deep discretized neural networks,” in *Advances in cryptology – CRYPTO 2018: 38th annual international cryptology conference, santa barbara, CA, USA, august 19–23, 2018, proceedings, part III*, vol. 10993, H. Shacham and A. Boldyreva, Eds. Cham: Springer International Publishing, 2018, pp. 483–512.

[68] A. Brutzkus, R. Gilad-Bachrach, and O. Elisha, “Low latency privacy preserving inference,” in *International Conference on Machine Learning*, 2019, pp. 812–821.

[69] J.-W. Lee *et al.*, “Privacy-preserving machine learning with fully homomorphic encryption for deep neural network,” *IEEE Access*, vol. 10, pp. 30039–30054, 2022.

[70] W. Han and K. E. Atkinson, *Theoretical numerical analysis: A functional analysis framework*. Springer, 2009.

[71] C. F. Dunkl and Y. Xu, *Orthogonal polynomials of several variables*. Cambridge: Cambridge University Press, 2014.

[72] J. Schlessman, “Approximation of the sigmoid function and its derivative using a minimax approach,” 2002.

[73] M. Chase *et al.*, “Security of homomorphic encryption,” *HomomorphicEncryption. org, Redmond WA, Tech. Rep*, 2017.