UNIVERSITY OF SURREY

UNIVERSITY OF SURREY

Faculty of Engineering and Physical Sciences

MSc Dissertation

Privacy Preserving Neural Networks

A project supervised by Professor Liqun Chen

Jamie C Dance (PG/T – Computer Science)

URN: 6661320

Abstract

This report delves into the advantages, disadvantages and design of implementing a Neural Network to work with encrypted data by using a modern technique called Homomorphic Encryption. The aim is to develop an insight into the framework which enables users to preserve their sensitive and confidential data whilst also being able to use predictive techniques such as Neural Networks. There are several real-world opportunities that could benefit from these techniques in delicate fields such as like Medical and Financial. By using these techniques, the data used can be held from dangerous third-party organisations and help to provide faster and more efficient developments in these fields.

Much of this project will include extensive research into Homomorphic encryption which makes this all possible. This includes all homomorphic encryption schemes and all previous cryptographic foundations and concepts previously published. We will explore the fundamental concepts of Fully Homomorphic Encryption, including practical implementations and development tools from Neural Networks. This topic is limited by its current literature but there has still be several suitable implementations in the recent years, a timeline has been built that showcases this work.

A comparison will be made into a few of the already proposed concepts published that tackle the concept of machine learning on encrypted data. This includes looking into each step of the implementation of a neural network; the encryption, the prediction model (a convolutional neural network, which is carried out using homomorphic evaluation functions), the masked prediction and finally the decryption by the user. Lastly an evaluation of the limitations and threats that this work can pose will be undertaken. This will include a detailed analysis of the challenges that may be faced, and the threats posed by the growing popularity of machine learning and more specifically predictive analysis.

Keywords: Machine Learning, Cryptography, Encryption, Homomorphic Encryption, Privacy-Preserving Machine Learning, Neural Networks.

Acknowledgements

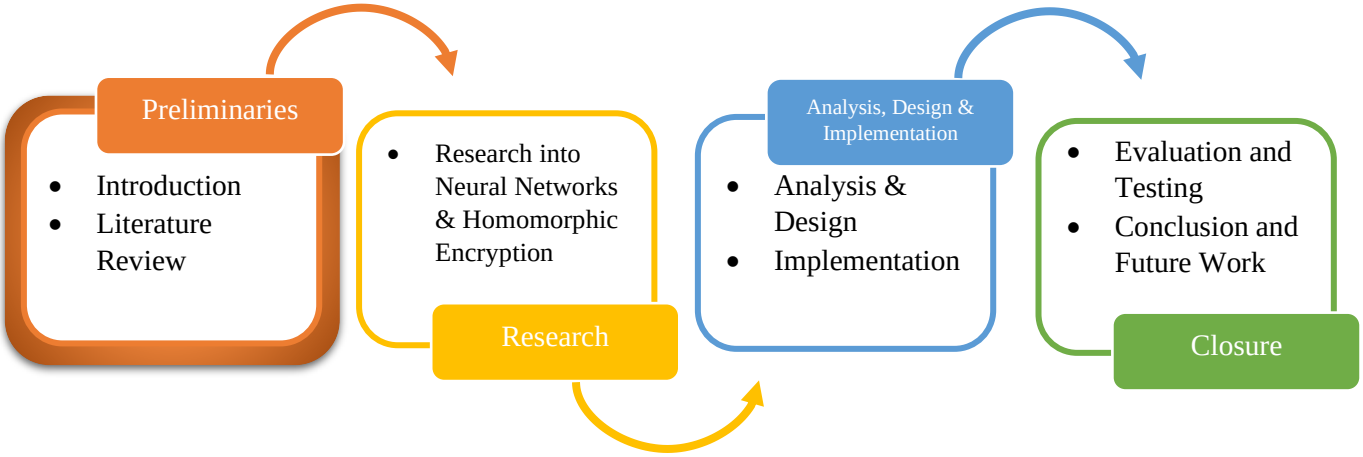
I would like to thank everyone at the University of Surrey for their commitment and help to not only me but all students. The last year has been tough due to COVID-19 but the university has done their best to adapt and evolve. In particular, I would like to thank Professor Liqun Chen for the advice and suggestions she has provided that allowed me to manage and complete this project.

I would like to thank my parents for their constant support and push which allowed me to get where I am today.

Finally, I want to thank my girlfriend Gemma Bayley for all the support he gave me during this difficult process and made me believe in myself.

Table of Contents

Part One: Preliminaries



Chapter 1:
Project Outline

1.1 Introduction

We live in the age of algorithms, a term not so long ago that was surrounded by puzzlement. With the rise of Machine Learning they are in every nook and cranny of civilization. They're not just a part of your cell phone, laptop or bank credit score, they are in your car, your house and even some of your fridges! Included within all that data can be very personal and sensitive data such as medical or financial data. So, in practice we have Data Scientists who are telling people to share their data to improve already existing mechanisms, whilst privacy experts are advising people to hide it or even delete it. This has led to the privacy dilemma: either sensitive user data must be revealed to the entity that evaluates the cognitive model (e.g, in the cloud), or the model itself must be revealed to the user so that the evaluation can take place locally (Florian Bourse, 2017). Applying machine learning to a problem that involves this data not only requires accurate predictions but also careful attention to maintaining data privacy and security.

This project will be focused on applying learned Neural Networks to encrypted data. This allows a data owner to send their data in an encrypted form to a cloud service that hosts the networks. The encryptions ensure the data remains confidential since the cloud does not have the means to decrypt it. Regardless, the cloud service will be capable of applying the neural networks to the encrypted data to make encrypted predictions that are then returned to the owner who can then decrypt them. That way the service provider does not gain any information about either the raw data or the predicted output.

The growing interest in Machine Learning as a Service (MLaaS), where a marketplace of predictors is available on a pay-per-use basis, requires attention to the security and privacy of this model. Not all data types are sensitive but in certain areas such as medical, financial and marketing MLaaS is becoming popular due to its versatility. Using the concepts in this report in the near future users will be able to securely analyse their data as a service online.

1.2 Aims and Motivation

As already mentioned in the previous section of this Chapter, this project aims to explore and compare the performance of some existing privacy-preserving neural network analytical tools. The motivation for the project came from my desire to explore new modern methods of applying machine learning rather than researching older techniques I have previously conducted on past projects. Initially I wanted to investigate into a modern Machine Learning technique used with Big Data, but my Supervisor Professor Liqun Chen specialised in Cryptography and recommend looking into some link between Machine Learning and Cryptography. After some early research I discovered a new area of work involving using Neural Networks on Encrypted data by large organisations such as Microsoft and Amazon. Prof. Liqun Chen suggested I undertook this project and explore potential improvements and future work.

The **first aim** of this project is to conduct a comprehensive study into this project's core concepts such as Homomorphic Encryption and Neural Networks. The aim is to thoroughly define the process, limitations and effects of using encrypted data with neural networks. In addition, we will discuss the how homomorphic encryption and more specifically fully homomorphic encryption provides the effective solution.

The **second aim** of this project is to research and explore some already existing methods of applying Neural Networks to Encrypted data that are available online such as CryptoNets. This will help us build an understanding as to how developers are tackling this problem currently. In addition, it with provide us with some background that might help us to improve our own model in the future.

The **third aim** is to explore and compare some already existing concepts by applying them to a common set of data. The goal will be to discuss the varying accuracy of these different methods and identify the strengths and limitations associated with them. It is important to note that different

systems/models may have different methods of evaluation which could cause comparisons to be difficult.

1.3 Project Objectives

In the previous section we have discussed the motivations and the aims for the project. In this section we will list the objectives for this project:

1. **Explore and understand some of the various elements within in the project such as Neural Networks, Encrypted data or more specifically homomorphic encrypted data.**

2. **Review and compare a few of the current solutions/systems**

3. **Develop a Privacy-Preserving Neural Network system**

1.4 Project Stakeholders?

The new methods implemented for the sake of this Project along with all its deliverables:

- Is the work of, and owned by Jamie Dance, student number 6661321. Jamie Dance is the Author of the Report and the developer of the new Applications of Neural Networks on Homomorphic Encrypted Data. Despite referencing throughout the project to a “We” this does not mean in any way that the Author had assistance or direct input by anyone other than himself or his supervisor.
- Is supervised by Professor Liqun Chen.

1.5 Project Scope and Context

In this section we will provide a better explanation of the overall scope of the project and we will discuss the importance of working with encrypted data.

Encryption is important because it helps protect information, confidential data, and can improve the security of communication between client applications and servers. Essentially, when your data is encrypted, even if unauthorized individuals or entities can access it, they will not be able to read it. According to data from the Software Alliance (Alliance, n.d.), cybercriminals stole 423 million identities in 2015 and 5 billion were exposed by data breaches in 2018. Cybersecurity threats are still multiple, and new additional threats and alternative methods are being developed to attract consumers of online services.

At the most basic level, encryption is very simple. The password replaces letters, numbers, and symbols with other characters to create a password. Encryption is a set of characters that replace the original data. The person who created the encryption has the key to decode it. The "key" is basically a number that describes the mathematical process of encryption. You can do this manually, which is very time consuming, or you can use a software solution to encrypt the data using an algorithm and create an encryption key. Without the key, computers or humans cannot read the actual data; if people without the key can access the data, it will be useless and useless. It is expected that users can decrypt or "decrypt" (as opposed to encryption) data using the encryption keys they generate. In this project we will discuss a new concept called “Fully Homomorphic Encryption”, which solves the issue of being able access to the data without actually ‘seeing’ the data.

1.6 Project Control and Risk Assessment

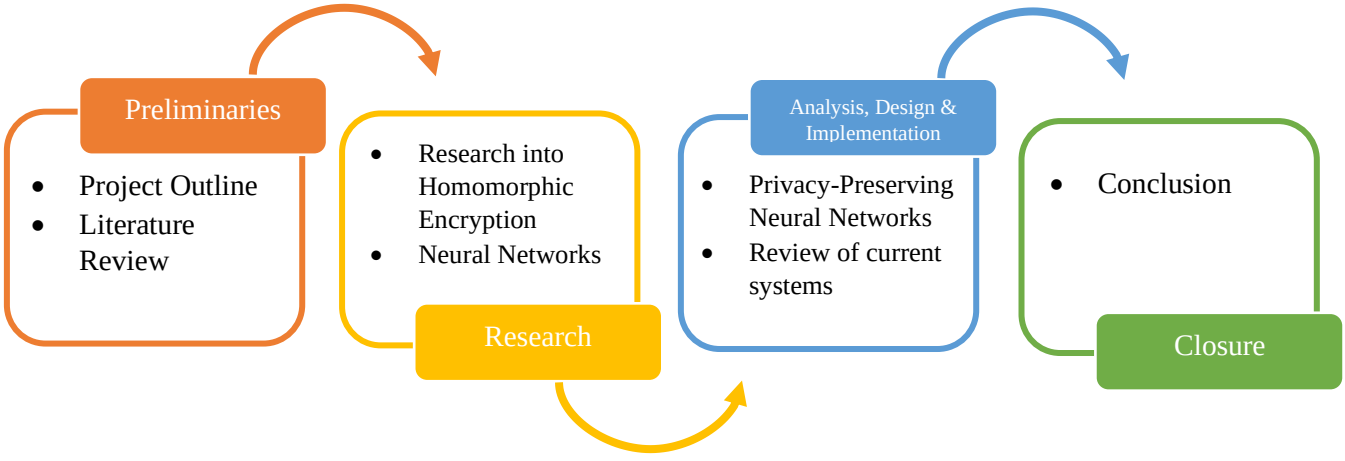
In order to keep control over the project several measures were kept in place to ensure that the project was completed smoothly and successfully in time. Many separate documents that will not be included in this report were created and used to assist in the completion of the project such as a project plan, project schedule and project notes. In addition, when necessary these notes were shared with Professor Liqun Chen for approval.

Various threats and risks have been taken in consideration in order to ensure that the project is completed successfully. These threats are identified and outlined below:

- The author of this project is experienced in Python, but several low-level concepts in this project must be developed in C++ which the author is not familiar with. This is a considerable task and there is significant task to perform.
- In addition to not being familiar with C++, this is the first time the author has explored into the world of cryptography. This is likely going to make research and individual development much harder.

1.7 Report Structure

Inspiration for the structure of the report has been taken from one of the sample dissertations provided. The report was written by Marios Erodotou on the Automated detection and tracking of Mycobacterium Tuberculosis cells (Erodotou, n.d.). The structure of the report allows for a clear and pleasant way for the reader to navigate whilst simultaneously allowing the author to be aware of the current progress against the project schedule. Therefore, I have decided to also split the report into **four** different sections which each represented with a different theme. Within each section are a number of relevant chapters which have a number of subsections within that chapter. For example, the current section is located within "Chapter One: Project Outline" which is within "Part One: Preliminaries". I have decided to use a colour theme for each part throughout the report to designate what content belongs to which section. As a result, any chapters or chapter subsections within a part will follow the same theme. For example, Part One follows an **Orange** theme and all following subsections also do. For each section there will be a brief introduction to the chapters that will follow, and also a summary at the end to provide a brief conclusion that sums up everything important that was discussed within the section. There will not be a summary for Part Four as the section itself is a conclusion of the project as a whole. For easier referencing I will also include an extra part at the end of each section that includes a list of corresponding references instead of one confusing block at the end of the report. A graphic will be included at the beginning of each section and will highlight the section and content that is to follow. Within each section of the graphic there is a summary of the significant chapters which allows to enhance the overall navigation and experience for the reader. The below graphic clearly illustrates the various sections and the lifecycle of this project:



The report contains a total of four sections and each section contains a number of chapters. Please find a brief description of each section below:

Part One: Preliminaries, the current part, makes an introduction to the overall project. It explains why this some basic concepts including how important the topic is. It also gives some background, including some related work, some of the motivations for the project and the project scope. Furthermore, the resources risks and objective of the project are outlined, and the structure of the

report is explained. Part One closes with Chapter Two, which reviews some literature relevant to the project.

Part Two: Research, in this part we will explore various methods already developed and published to help us understand some of the possible approaches to the problem. This part should be used by the reader as a reference consistently in the following sections.

Part Three: Analysis, Design & Implementation, in this section we will describe the Analysis, Design and Implementation of our developed system. In this section we will also include all ideas considered in the design process and will incorporate a description of some of the development used for the system. It will conclude by describing the implementation stage of the project and describe the final system.

Part Four: Closure, this part concludes the report by evaluating the overall report by testing the final system. This part will also outline what went well during the project and what could be have been improved. We will discuss what could have been done differently during the project in regard to avoiding some of the pit falls that were experienced. Lastly, recommendations for how the actual system could be improved and what potential work could be done to further evolve this topic of research.

1.8 Measures of Project Success

This project will be a success if everything defined in the project objectives (subsection 1.3) is achieved. In case the project has achieved part but not all the specification then it will be considered as incomplete but not failed. It is important to note that this field of study is still very new and is mostly still in research stage for most organisations. Therefore, the author is not expected to create a perfect system but rather attempt to improve already existing system.

Chapter 2:
Literature Review

2.1 The Problem Background

In this section, some insight will be given into some of the concepts included in this report in hopes of presenting a more detailed representation of what stage this current work is at and how it has developed over the years.

As mentioned in the Introduction, Machine Learning is a quickly developing and very powerful tool for people across the globe. But with growing success comes increasing issues and one of them issues is the topic of privacy and cybersecurity. Data scientists across the globe are encouraging people to share their sensitive material to create important developments into fields such as medicine although people aren't prepared to share information that could be detrimental for themselves. Typically for one to analyse a set of data it is important to know the input, the output and the scope/aim of the data. That cannot be the case with encrypted data as it is not possible for the analyst to see the content of the data at all. By using homomorphic encryption, we can permit users to perform computations on encrypted data without decrypting it. Processing large amounts of encrypted data can be very difficult as the processing power needed to apply machine learning techniques is significantly higher.

2.2 Recent Advances

The combination of Neural Networks and more generally Machine learning with encrypted data is a modern-day concept.

In traditional cloud storage and computation solutions the cloud needs to have unencrypted access to the customers data to compute on it, necessarily exposing the data to the cloud operators. Customers need to trust the service provider to store and manage their data appropriately, e.g., not share it with third parties without the customers' consent. As a result, data privacy relies on access control policies (such as an access control list) implemented by the cloud and trusted by the customer. With the advances in homomorphic encryption technology, it is possible to allow computations to be performed directly on encrypted data. In recent years, Fully Homomorphic Encryption has demonstrated exceptional development progress. Although, the current literature is extremely restricted by practitioners searching for suitable implementations. What is meant by this is that the content is not thorough or complete enough. In this report the focus will solely be on neural networks used within a homomorphic encryption cryptosystem. We will focus on how specifically neural networks have developed over the recent years to use homomorphic encryption, identify the current solutions, the problems, potential pit-falls and the possible future developments.

The diagram below shows an example of a basic example of a traditional method of encryption (1) that requires the user to apply some function (addition in this example) to two numbers x_1 and x_2 then encrypt them. This is something that can be done with any encryption, but with Homomorphic Encryption you can go the other way around (2), you can first encrypt then apply some function. This can be done without knowing the value of x_1 or x_2 and without knowing the result. In this example we have used addition but the same can be done with multiplication and a multitude of functions.

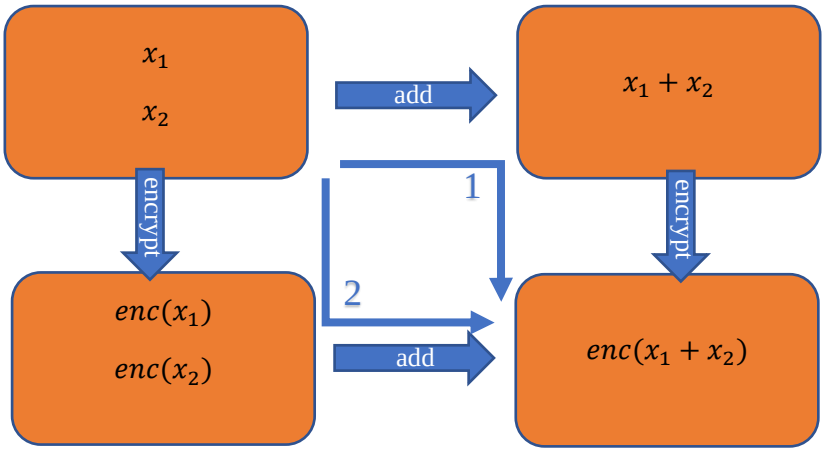


Figure 1: Basic example of 1) original encryption and 2) homomorphic encryption concept

2.3 Related Work

2.3.1 MLaaS with Homomorphic Encryption

In this paper we are mostly focused on neural networks, their benefits and how they are effective with homomorphic encryption. Although the future of homomorphic encryption is not only privacy-preserving neural networks but privacy preserving **machine learning**. This not only includes neural networks but all forms of machine learning and predictive tools such as Decision Trees or Naïve Bayes. The term MLaaS is the appreciation of "Machine Learning as a Service", it refers to providing all machine learning services in a cloud computing platform. In the future this will act as a flexible and scalable solution for users to train models and predict information remotely. Naturally, something this convenient comes with its problems, and the most serious problem is online security and privacy concerns (E.Witchel, 2018). The problem lies with making predictions and classification models with hugely sensitive data. This is usually medical, financial and sometimes advertising data, although there is examples of others. Homomorphic Encryption provides a graceful solution to the problem of security in the cloud. It allows users to send encrypted data (data that cannot be ready by anyone other than the user allows to read) into a remote server where a third-party blindly (does not learning anything about the input or output of the data) processes the data and returns the solution completely unaware of who or what data was involved in building the system.

2.3.2 Homomorphic Encryption for Privacy Friendly Machine Learning

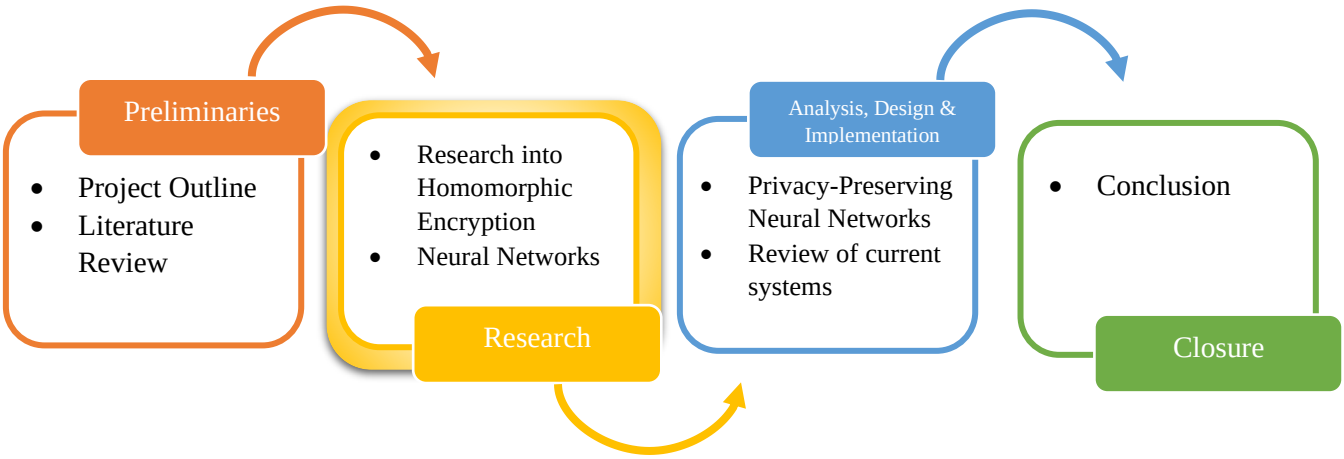
Before the arrival of Homomorphic Encryption, the common approach to protect sensitive or confidential data was to use Secure Multi Party Computation (SMC). This involves both parties involved to create a communication protocol, as shown in the paper (Zheng Q, 2018). This technique was combined with regular encryption and were called oblivious neural networks. The authors in (Goldreich, 1998)recommended one scheme where the original user/owner of the data encrypts their data using original encryption and sends it to the prediction service on the cloud and the service returns the encrypted prediction, which the user then decrypts, administers non-linear activation functions, encrypts, and returns to the server once again. This would continue until every layer of the predictive system was calculated. Although, this method was found to leak a lot of information (M. Barni, 2006), hence making it particularly unreliable to users as it could result in data theft. As a result, it could not be commonly used for important and confidential data as it was originally intended was not used until it was refined into fully homomorphic encryption as is commonly used now.

Summary of Part One

Part one has been constructed to serve as an informative introduction to the report that clearly lays out some of the initial key concepts of the report and creates a foundation to be developed upon further into the report. During this part we went through chapters One and Two which were composed of an introduction and a literature review. In the first chapter, an overview of the whole project was given that outlines all the key factors as to how the report was developed and what was intended to be provided to the reader by the author. This included the project's aims, motivations, objectives and stakeholders. In addition, it listed the resources used and defined the measures of project success. In Chapter Two, a literature review was started in which some initial conclusions were made for the report and the problem background was described. Within this section we described the need for reliable confidential data analytics and the difficulty that lies within it. In addition, we spoke of a couple related works to machine learning as a united concept with MLaaS, which is the future of homomorphic encryption.

It is important to note that not everything was covered in this initial literature review. The next section should be used as an extension of the previous literature review.

Part Two: Research



Chapter 3: Research into Homomorphic Encryption

3.1 Research into Privacy-preserving neural networks with Homomorphic Encryption

Modernized encryption techniques are built to enforce high levels of security and are widly considered the best solution to protection sensitive data when it is required to be process by some third-party. Regular encryption techniques fall into the problem of data vulnerability as the date must be processed or prepared. Fully Homomorphic Encryption is considered the holy grail of cryptography and to many the elusive goal that could solve all contemporary cybersecurity problems (V, 2011) (C, 2009) (Gentry C, 2011). Within this section we will discuss the fundamental concepts of Fully Homomorphic Encryption, practical implementations, state-of-the approaches, limitations, advantages, disadvantages, potential applications, and development tools focusing on neural networks.

Furthermore, this section is also meant to be used as a reference for the following chapters at the Analysis (Part Three) phase of the project. This chapter is meant as more of a guidance or general explanation of some of the core concepts to avoid confusion and to allow the reader to have a grasp some of the algorithms used.

3.2 What is Encryption?

To prevent forgery and to ensure the confidentiality of the content of the letter, for centuries the sender of the letter usually signed the letter, then sealed it in an envelope, and then delivered it to the sender. Public key cryptography was discovered nearly five years ago (Goldreich, 1998)and it has completely changed the way people communicate securely and authentically. People can now communicate with each other or send information in a secure and authenticated manner over open and insecure networks. In doing so, the same two-step method was followed. That is to say, before sending the message/data, the sender of the message/data will use a digital signature scheme to sign it, and then use the private key encryption algorithm to perform the message/data (and signature) under a randomly selected encryption key Encryption. The receiver's public key will then be used to encrypt the random encryption key. This is known as the two-step approach signature-then-encryption (M. Barni, 2006).

In public key cryptography we use two keys, one **public** and one **private**, related in a mathematical way. The public key can be published in a directory along with the user's name. Anyone who then wishes to send a file to the holder of the associated private key will take the public key, encrypt a message under it and send it to that key holder. The idea is that only the holder of the private key will be able to decrypt the message. Below shows an example of how the process works:

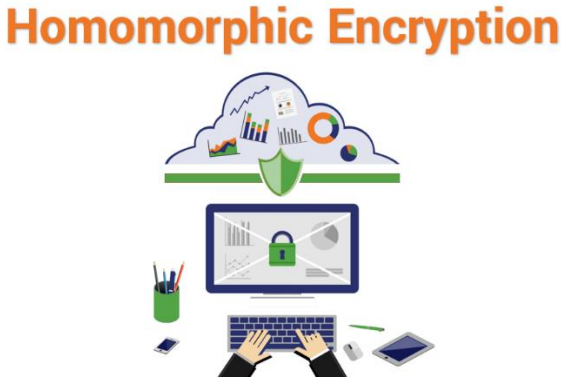
File + Bob's public key = Ciphertext,
Ciphertext + Bob's private key = File

Hence anyone with Bob's public key can send Bob a secret message. But only Bob can decrypt the message, since only Bob has the corresponding private key.
Public keys systems work because the two keys are linked in a mathematical way, such that knowing the public key does not allow you to compute anything about the private key. But knowing the private key allows you to unlock information encrypted with the public key.

The concept of being able being able to encrypt using a key which is not kept secret was so strange it was not until 1976 that anyone thought of it. The idea was first presented in the seminal paper of Diffie and Hellman named *New Directions in Cryptography* (Diffie, 1976). It was not until a year or so later that the first and most successful system, namely RSA (Rivest-Shamir-Adleman), was invented.

3.3 Homomorphic Encryption

This section discusses the basic concept and evolution of homomorphic encryption based on the representative work in this field.



Source: https://www.thesslstore.com/blog/what-is-homomorphic-encryption/

In the field of cryptography, the term homomorphic encryption defines an encryption system that can perform certain computational functions on ciphertext. The output has the characteristics of the function and the input format. The system cannot access the information about the ciphertext and keys. It only uses publicly available information and there is no risk of data leakage. As mentioned in the previous section, the concept of homomorphic encryption refers to the mapping between functions in the message space and the encrypted text space. The homomorphic function applied to the ciphertext provides the same result (decryption) as the function applied to the original unencrypted data.

Let m_1 and m_2 be messages, c_1 and c_2 be their corresponding ciphertexts. The operation $\ddot{+}$ in an additively homomorphic encryption produces the ciphertext $c_+ \leftarrow c_1 \ddot{+} c_2$ that can be decrypted to $m_1 + m_2$.
Similarly, for $\ddot{\times}$ in a multiplicatively homomorphic encryption, it generates the ciphertext $c_\times \leftarrow c_1 \ddot{\times} c_2$ that is decrypted to $m_1 \times m_2$. Both encryptions obtain ciphertexts c_+ and $c_\times$, without knowing m_1 and m_2.

There are three types of homomorphic encryption cryptosystems: Partially Homomorphic Encryption, Somewhat Homomorphic Encryption and Fully Homomorphic Encryption. In the following subsections we will discuss their limitations and scopes.

3.2.1 Partially Homomorphic Encryption
Partially Homomorphic Encryption supports an unlimited number of operations. For example, additive homomorphic encryption allows an unlimited number of additions but does not allow multiplication.
Ronald Rivest, Adi Shamir, and Leonard Adleman (RSA) cryptosystem are the first multiplicative partially homomorphic encryption (Rivest R, 1978). In general, given two messages m_1 and m_2 and their respective ciphertexts $c_1 = (m_1^e) mod\ n$ and $c_2 = (m_2^e) mod\ n$, where e is chosen such that $\gcd(e, \emptyset) = 1$ for $\emptyset = (q_1 - 1) \cdot (q_2 - 1)$ with large primes q_1 and q_2, and $n = q_1 \cdot q_2$. The ciphertext with the product of the original plaintexts is computed as

$$c_\times \leftarrow (m_1 \cdot m_2)^e\ mod\ n = (m_1^e)\ mod\ n \cdot (m_2^e)\ mod\ n$$
$$= c_1 \ddot{\times} c_2$$

Due to its deterministic encryption algorithm, RSA is not semantically secure. Taher El-Gamal is another relevant multiplicative partially homomorphic encryption (T, 1985).

The Shafi Goldwasser and Silvio Micali (GM) cryptosystems are the first additively partially homomorphic encryption cryptosystems (Goldwasser S, 1982). According to GM cryptosystem, there are two messages m_1 and m_2 and their respective ciphertexts $c_1 = (b_1^2 \cdot e^{m_1}) mod\ n$ and $c_2 = (b_2^2 \cdot e^{m_2}) mod\ n$, where b_1^2 and b_2^2 are quadratic nonresidue values such that $\gcd(b_1^2, n) =$

$\gcd(b_2^2, n) = 1$, and e is one of the quadratic nonresidue modulo n values with $\left(\frac{x}{n}\right) = 1$.
The GM scheme has a homomorphic property, where the encryption of $m_1 + m_2$, is

$$c_+ \leftarrow [(b_1 \cdot b_2)^2 \cdot e^{m_1+m_2}] \, mod \, n$$
$$= [(b_1^2 \cdot e^{m_1}) \cdot (b_2^2 \cdot e^{m_2})] \, mod \, n$$
$$= (b_1^2 \cdot e^{m_1}) \, mod \, n \cdot (b_2^2 \cdot e^{m_2}) \, mod \, n = c_1 \ddot{+} c_2$$

However, GM is not an efficient solution because the ciphertext can be hundreds of times larger than the original plaintext.
The related additional partially homomorphic encryption cryptosystem was invented and named by Josh (Cohen) Benaloh, 1994 (J, 1994), David Naccache and Jacques Stern (NS), 1997 (Naccache D, 1998), Tatsuaki Okamoto and Shigenori Uchiyama (OU), 1998 (Okamoto T, 1998), Pascal Paillier, 1999 (P, 1999), Ivan Damgård and Mads Jurik (DJ), 2001 (Damgård I, 2001), Steven Galbraith, 2002 (SD, 2002) and Akinori Kawachi, Keisuke Tanaka and Keita Xagawa (KTX), 2007 (Kawachi A, 2007).

The encryption process in partially homomorphic encryption does not guarantee a certain level of security. The worst-case severity of the "noisy" problem is a direction of the security solution. The term noise means that an adequate amount of error is injected into the encrypted message and generates an inaccurate relationship (M, 2018).

3.2.2 Somewhat Homomorphic Encryption
A certain degree of homomorphic encryption supports a predetermined number of different homomorphic operations, thereby limiting the number of allowed operations. Every transaction adds potential noise, so your correct assessment depends on only taking a limited number of actions. When the noise exceeds a certain threshold, the message decryption fails.
The Dan Boneh, EuJin Goh, and Kobbi Nissim (BGN) scheme (Boneh D, 2005) is the first method to allow addition and multiplication and fixed size ciphertext. The hardness of BGN is based on a subgroup decision problem (K, 2004), which decides whether an element is a member of a subgroup G_p of group G of order $n = q_1 \cdot q_2$. In BGN, ciphertexts $c_1 = g^{m_1} \cdot h^{e_1}$ and $c_2 = g^{m_2} \cdot h^{e_2}$ encrypts m_1 and m_2 messages, where g and u are two random generators from G, $h = u^{q_2}$ is a random generator of the subgroup of G of order q_1 and random numbers e_1 and e_2 from the set $\{0,1, \dots, n-1\}$.
The encryption of $m_1 + m_2$ is computed as:

$$c_+ \leftarrow g^{m_1+m_2} \cdot h^{e_1+e_2+e} = (g^{m_1} \cdot h^{e_1}) \cdot (g^{m_2} \cdot h^{e_2}) \cdot h^e$$
$$= c_1 \cdot c_2 \cdot h^e = c_1 \ddot{+} c_2$$

Nonetheless, BGN is impractical due to it computing c_+ only one using the bilinear map property, which maps $s : G \times G = G_1$, where G_1 is a group of order $n = q_1 \cdot q_2$.
Let $g_1 = s(g, g)$ and $h_1 = s(g, h)$, where g_1 is of order n and h_1 is of order q_1. Thus, there is α such that $h = g^{\alpha q_2}$.
The encryption of $m_1 \cdot m_2$ is computed as:

$$C_\times \leftarrow g_1^{m_1 m_2} \cdot h_1^{m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2 + e}$$
$$= g_1^{m_1 m_2} \cdot h_1^{m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2} \cdot h_1^e$$
$$= g_1^{m_1} \cdot g_1^{\alpha q_2 (m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2 + e)} \cdot h_1^e$$
$$= g_1^{m_1 m_2 + \alpha q_2 (m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2 + e)} \cdot h_1^e$$
$$= s(g, g)^{(m_1 + \alpha q_2 e_1)(m_2 \alpha q_2 e_2)} \cdot h_1^e$$
$$= s(g^{m1 + \alpha q_2 e_1}, g^{m_2 + \alpha q_2 e_2}) \cdot h_1^e$$
$$= s(g^{m_1} \cdot g^{\alpha q_2 e_1}, g^{m_2} \cdot g^{\alpha q_2 e_2}) \cdot h_1^e$$

$$= s(g^{m_1} \cdot h^{e_1}, g^{m_2} \cdot h^{e_2}) \cdot h_1^e$$
$$= s(c_1, c_2) \cdot h_1^e = c_1 \ddot{\times} c_2$$

where $m_1 e_2 + e_2 m_1 + \alpha q_2 e_1 e_2 + e$ is uniformly distributed in $\mathbb{Z}_N$, and $c_\times$ is uniformly distributed exncryption of $(m_1 \cdot m_2) mod\ n$, but now in G_1 rather than G. However, BGN is still additively homomorphic in G_1.

3.2.3 Fully Homomorphic Encryption concept

After 30 years of development in this field, Gentry (Goldwasser S, 1982) produced the first fully homomorphic concept in 2006. The field began with the invention of public keys (Didie W, 1976). In his concept, based on the Ideal Coset problem, he produced a somewhat homomorphic guided cipher with additional hardness. This method can perform homomorphic evaluation of the decryption function. The construction of a somewhat homomorphic cipher uses concepts from lattice algebra. The idea I in the ring $\frac{Z[x]}{f(x)}$ with $f(x)$ of degree n satisfies $a + b \in I$ and $\in \frac{Z[x]}{f(x)}$.

The scheme encrypts plaintexts m_1 and m_2 in ciphertexts $c_1 = (\varphi_1) mod\ B_J^{pk}$ and $c_2 = (\varphi_2) mod\ B_J^{pk}$, where $\varphi_1 \leftarrow samp(B_I, m_1)$ and $\varphi_2 \leftarrow samp(B_I, m_2)$ samples from the coset $I + m_1$ and $I + m_2$, respectively, and B_J^{pk} defines a secret base for some idea J in a ring R with a basis B_I of I, for relative primes I and J.

Encryption of $m_1 + m_2$ is computed as

$$c_+ \leftarrow (\varphi_1 + \varphi_2) mod\ B_J^{pk} = (\varphi_1) mod\ B_J^{pk} + (\varphi_2) mod\ B_J^{pk}$$
$$= c_1 \ddot{+} c_2$$

and $m_1 \times m_2$ is computed as

$$c_\times \leftarrow (\varphi_1 \cdot \varphi_2) mod\ B_J^{pk} = (\varphi_1) mod\ B_J^{pk} \cdot (\varphi_2) mod\ B_J^{pk}$$
$$= c_1 \ddot{\times} c_2$$

The bootstrapping process reduces noise in the ciphertext and can be applied an unlimited number of times. Therefore, both aspects allow the construction of the first Fully Homomorphic solution.

Gentry's lattice Fully Homomorphic Encryption approach had potential, but it also had several issues. The extremely large computational cost and intense implementation made it impractical. Ten years later, fully homomorphic cryptographic encryption clusters have developed into four main areas: Ideal Lattice-based, Over Integers, Error-Based Learning, and N-th Truncated Polynomial Ring unit (NTRU)-based.

The first area's work follows Gentry's original idea, and its hardness was based on the problem of lattice reduction. The second refers to the integer-based method (Van Dijk M, 2010) (Brakerski Z, 2011), where the difficulty of the scheme is based on the approximate greatest common divisor (AGCD) problem (C, 2010). The third series includes solutions based on Error With Learning (LWE) (Gentry C, 2013) and Ring Learning with Error (RLWE) (Brakerski Z, 2012) (Fan J, 2012) (Z, 2012). Lastly the fourth family is of NTRU (Rohloff K, 2014) and subsequent works (Hiromasa R, 2015) (Alperin-Sheriff J, 2014), which are also based on the lattice problem.

3.4 Fully Homomorphic Encryption

The emergence of the first fully homomorphic encryption scheme has a major impact on the design of a more secure system but has no impact on its implementation. The high level of security of a fully homomorphic encryption solution can improve many technologies, such as outsourcing computing in a cloud environment. However, due to some limitations, the effective implementation of fully homomorphic encryption still has a long way to go. This section introduces the formal definition and basic concepts of fully homomorphic encryption, such as bootstrapping and key switching.

3.2.4 Timeline

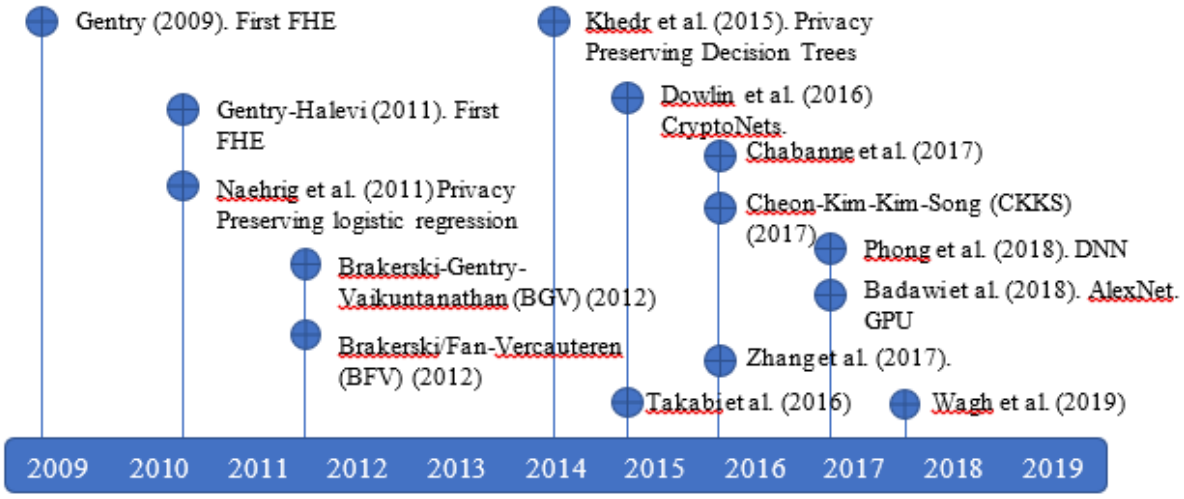


Figure 2: Timeline of the most relevant inventions in the history of Homomorphic Encryption

Rivest (Rivest RL, 1978) introduced the term privacy homomorphism to formally describe fully homomorphic encryption, the main idea is to perform arbitrary calculations on encrypted data without using decryption keys. The general idea behind fully homomorphic encryption is that the function f can be effectively expressed as a circuit for processing encrypted homomorphic data, such as programs, math operations, etc. (C, 2009). Fully homomorphic encryption is considered a promising post-quantum tool (R, 2017). Today's public key cryptography relies on the difficulty of solving problems, such as factoring or discrete logarithms. These widely studied problems are considered difficult to solve in classical computers.

However, alternative people equipped with sufficiently large quantum computers can easily solve them. Although quantum computers do not exist today, their potential is considered a threat. The essence of fully homomorphic encryption is to produce an output of ciphertext $f(c)$ for any desired function f and ciphertext message c of plaintext m, as long as the information of c, $f(c)$ and m cannot be exposed. The function f can be calculated efficiently. The figure in the following subsection illustrates the expected performance of the fully homomorphic encryption scheme ε as a simple black box model in computer systems. The challenging task is to find the right mechanism, $Evaluate_\varepsilon$, that successfully leads to the correct output within a reasonable time.

The following sections clarify concepts such as $Encrypt_\varepsilon$ and $Evaluate_\varepsilon$ and describe other basic elements, such as bootstrapping and key-switching.

Notation

Formally, the fully homomorphic ε encryption scheme defines a traditional public key scheme with four operations: $KeyGen_\varepsilon$, $Encrypt_\varepsilon$, $Decrypt_\varepsilon$, and $Evaluate_\varepsilon$ (C, 2009). The computational complexity of all operations must be a polynomial about the safety parameter λ, where:

- $KeyGen_\varepsilon$ takes λ as input and generates a public key pk and a secret key sk as output, where pk is mapped from the plaintext space $\mathbb{P}$ to the ciphertext space $\mathbb{C}$ and sk in the opposite direction.
- $Encrypt_\varepsilon$ uses pk and plaintext $m \in \mathbb{P}$ as input, and generates ciphertext $c \in \mathbb{C}$ as output.
- $Decrypt_\varepsilon$ defines the reverse process of $Encrypt_\varepsilon$. It receives sk and $c \in \mathbb{C}$ as input and outputs plaintext $m \in \mathbb{P}$.
- $Evaluate_\varepsilon$ will input pk, circuit $\delta \in \delta_\varepsilon$ and ciphertext tuple $C = < c1, \ldots, ct >$ to encrypt $M = < m1, \ldots, mt >$ for the input line of δ; generates the ciphertext $C' \in \mathbb{C}$ such that $Decrypt_\varepsilon\ (sk, C') = \delta(M)$.

Therefore, given the encrypted C of M, the required function of the $Evaluate_\varepsilon$ operation is to obtain the ciphertext $C' \leftarrow Evaluate_\varepsilon\ (pk, \delta, C)$ to encrypt $\delta(M)$ under pk, where $\delta(M)$ defines δ in the output unencrypted message M on pk. Furthermore, correctness and compactness are the basis for the formal definition of the fully homomorphic encryption scheme. They can be represented using the four basic operations defined above.

Definition 3. Fully Homomorphic Encryption
A the homomorphic encryption scheme ε evaluates all circuits compactly, it is fully homomorphic, namely:

$$Decrypt_\varepsilon\big(sk, Evaluate_\varepsilon(pk, \delta, C)\big) = \delta(M)$$

Since the first fully homomorphic encryption scheme, the term bootstrapping or "re-encryption" has been cornerstone. The next section will emphasize its importance and describe the process.

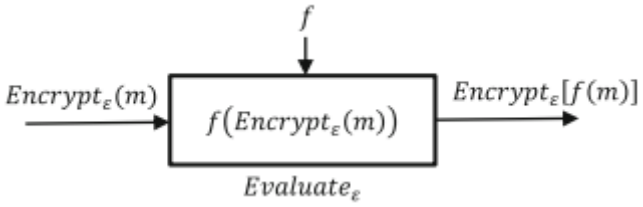


Figure 3: Homomorphic Encryption Theory

Bootstrapping
The security of the fully homomorphic encryption scheme lies in hiding the original message with a certain degree of noise. Before the introduction of fully homomorphic encryption, noise could only be removed by decryption; the number of operations on the ciphertext was limited. The error increases with each homomorphic operation, and when the error reaches the threshold, the decryption process is useless.
The concept of bootstrapping is introduced to keep the error below the threshold. It allows you to create the first fully homomorphic encryption scheme based on a somewhat homomorphic encryption scheme that can be started, that is, a scheme that can homomorphically evaluate its own decryption function. The recrypt function is the central part of the bootstrapping and is used to reduce noise in the ciphertext. It can be used an unlimited number of times to obtain new encrypted texts. Therefore, the re-encryption operation can ensure that the ciphertext is correctly decrypted after an unlimited number of operations.
Typically, the recrypt function re-encrypts the ciphertext (the plaintext is now double encrypted), removing the internal encryption using the cipher key (Armknecht F, 2015) to homomorphically evaluate the double ciphertext.

Key-switching
During the bootstrapping process, the second secret key is the basis for homomorphic encryption / decryption of the ciphertext. In terms of algorithms, bootstrapping can be defined as:

$$C' \leftarrow Encrypt_\varepsilon(pk_2, Decrypt_\varepsilon(sk_1, C))$$

where the new ciphertext C' contains less noise than the original ciphertext C. sk_1 and C are encrypted under the public key pk_1, and C' is encrypted under pk_2. The sk_1 encryption is usually referred to as the bk activation key.

The selection of keys is the basis for the correct operation of the process; the quality of the selection and development of keys is directly proportional to the performance of the bootstrapping execution. There are two options for defining bk: Encrypt the key sk1 $Encrypt_\varepsilon(pk_1, sk_1)$under itself, or another key $Encrypt_\varepsilon(pk_2, sk_1)$. In the self-encrypting key sk_1, the ciphertexts C and $\boldsymbol{C'}$ are

encrypted under the same key, so the cycle security [33] avoids the use of multiple keys. In contrast, the advantage of the key change alternative is that it does not require round-robin security but must handle multiple keys. A key limitation of the key-switching method is the number of keys available, that is, n keys can only achieve hierarchical homomorphism, because they allow n bootstrapping operations to be performed.

The main disadvantage of the bootstrap method is the calculation cost. Indirect costs become the main disadvantage of all fully homomorphic encryption practicality. Most start-up routines are complex and time-consuming. Even with these limitations, researchers are still trying to solve these shortcomings through high-performance, distributed, and parallel computing technologies.

Chapter 4: Research into Neural Networks

We already know that machine learning is a process of creating automated analysis of data and building analytical models. Typically, there are three different types of machine learning methods; Reinforcement Learning, this is where the environment interacts with the algorithm through a process of actions, in which it receives penalties or rewards (P, 1999), hence the name *reinforcement*. This model is attempts to maximise the cumulative reward.
Unsupervised Learning, this is where the algorithm attempts to find patterns in a given set of data, without any prior knowledge to assist it (Damgård I, 2001).
Supervised learning, this is where the model tries to figure out the mapping between input and output. In general, this means that you build a model by observing examples of input data that already have corresponding labelled output data. The result can be categorical (Classification) value or real valued (regression) (Vapnik, 1999).
At a high level, a neural network is a computational model based on a set of simple processing units, which communicate with each other by sending signals through a large number of weighted connections (Schmidhuber, 2015). The important components of the neural networks are:

- A group of processing units, called neurons or nodes.
- Connections between the units. They represent functional dependencies (that is, the edges from x to y indicate that y is a function of x). Usually, each connection is associated with a "weight".
- External inputs "bias/offset" input for each unit.
- Propagation rule, which determine the effective output of a node based on the internal and external inputs of the node.

The architecture of the neural network is nothing more than a systematic arrangement of layers. A layer is a collection of processing units (nodes). Each node does simple work, receiving inputs from other nodes (and external deviations) to calculate the output that propagates to other nodes. There are three types of layers: the nodes in the input layer in this layer only receive input on the network, and the nodes in this layer in the output layer send the results outside the network. In addition to the input and output layers, for the neural network to be practical, there must be at least one intermediate layer. These are called hidden layers and they take information from them and provide output to the layers of the network. In most cases, the propagation rule used in the network is a standard weighted sum, where each node provides an additional contribution, weighted by the connections of the nodes connected to it. In the standard neural network literature, the output of each node is called "activation." Each node applies some nonlinear function to the weighted sum of the inputs from other nodes, and the result is called the activation of that node.

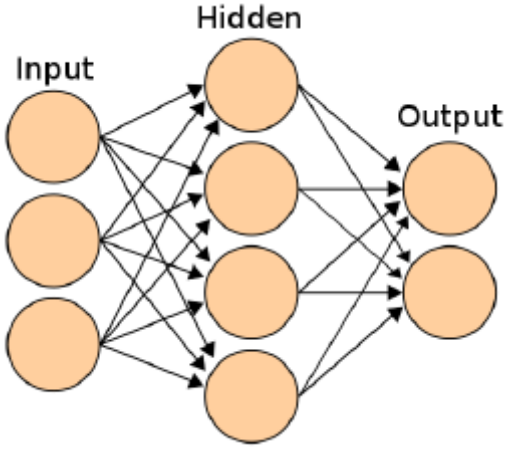


Figure 4: Diagram of a three-layer Neural Network

4.1 Convolutional Neural Networks

The data used later in this report to compare systems is the MNIST dataset. It contains the several handwritten digits ranging from 0-9. Although the design of the input and output layers of the network is quite simple. For example, since the input image is composed of 28 × 28 pixel values, the input layer will contain 28 * 28 nodes. Since the task is to classify numbers, the output layer will consist of 10 nodes, each representing a single number (0-9). The value in the output node determines which number is the winner. But the hidden layer design is usually challenging and requires a lot of research. The handwritten image recognition task itself has been solved with very high accuracy using the convolutional layer in the network. These neural networks with convolutional layers after the input layer are called Convolutional Neural Networks (CNN).

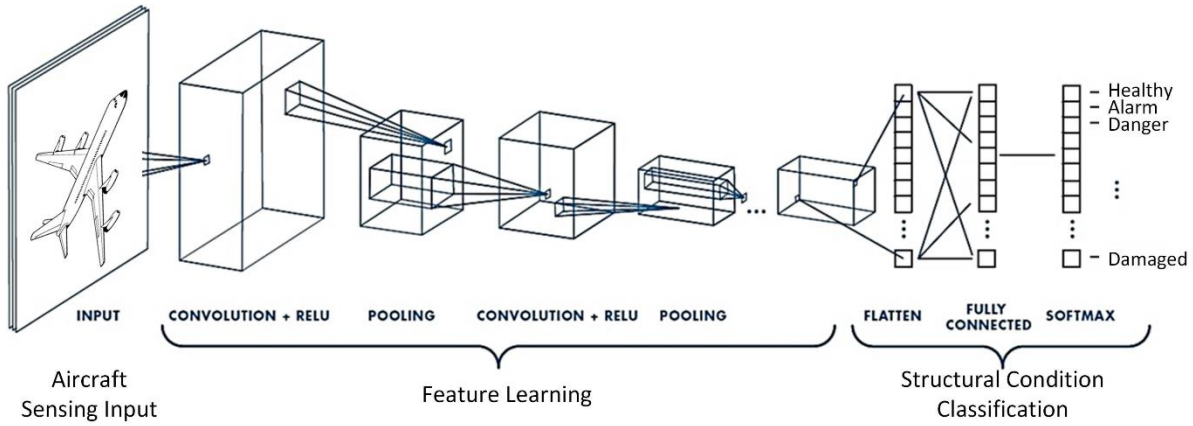


Figure 5: A sample Convolutional Neural Network architecture

Source: A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. Retrieved from: (Tabian, et al., 2019).

They have been shown to be effective in image classification because, unlike standard networks, CNN's can capture spatial topology well. This means that if the standard network is trained on a fixed layout data set where all input pixels are the original data set, the result will be the same (Y. LeCun, 1998). The following sections show some of the theoretical aspects of CNN's at a high level.

4.2 Layers

4.2.1 Convolution Layer

The main purpose of this layer is to extract entities from the input. The input is convolved with the filter (or kernel) to generate a feature map (convolution output). Convolution is performed by sliding a filter on the input. The filter is just a matrix of weights and strides. The stride is the step size of each movement of the convolution. (That is, if the stride is 1, the filter slides pixel by pixel.) At each filter stop, matrix multiplication is performed. Then summarize the results on the feature map. Multiple filters can be used to perform multiple convolutions on a single input. It helps to extract multiple features from the input. You can add pixels of zero value (or the same value as the border) around the input image so that the size of the feature map is no smaller than that of the input. This is called padding. This ensures that the kernel and the convolution step with input give the same size output. This layer involves computing a large number of matrix multiplications (also dependent on input size, filter size, and step size), making it the most computationally expensive layer. When using CNN, the important hyperparameters to choose from are:

- Kernel size.
- Filter count.
- Stride.

- Padding.

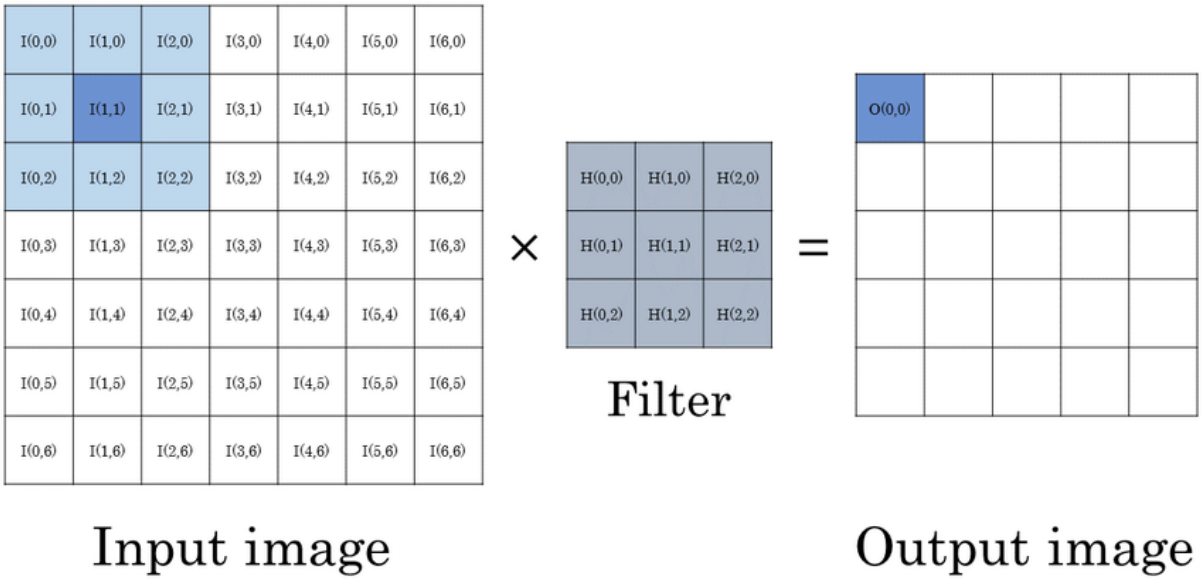


Figure 6: An example illustration of convolution

Source: Streaming Architecture for Large-Scale Quantized Neural Networks on an FPGA-Based Dataflow Platform
Retrieved from: (Baskin, 2017)

4.2.2 Activation Layer

The convolutional layer and the fully connected layer are basically weighted multiplication followed by offset addition. So, in these layers, all operations are linear. (The form is $w * x + b$). In order to introduce nonlinearity in the network, an activation layer is used. These layers have no variables such as weights or biases, they have only one function. The neurons in this layer apply a single function to the input and pass the result to the next layer. The activation function is usually sigmoid, hyperbolic tangent (tanh), modified linear operation (RelU) and other functions. If there is no nonlinear activation function, even if there are many layers in the network, the sum of these layers will only give a linear function, so it cannot be used to create an effective model for nonlinearly separated data. However, when it comes to encrypted values, due to the limitations introduced by homomorphic evaluation (described in the previous section), ReLU or Tanh or other commonly used trigger functions cannot be used. Instead, you can use polynomial approximations of these functions or other nonlinear polynomials suitable for the use case.

4.2.3 Pooling Layer

Typically, a pooling layer is added between convolutions to achieve the purpose of reducing resolution. (Reduce dimensionality). This helps reduce the number of parameters (weights and biases) and shortens the training time. They also help control overfitting to some extent. The most popular types of grouping are maximum grouping and average grouping, as shown in Figure x. In Max-Pooling, only the maximum value will be generated in the window and other values will be discarded. In grouping averages (mean), the average of all values in the window is used as the output. Max-Pooling is effective in extracting the sharpest and most important features, while averaging can facilitate extraction (A. Krizhevsky, 2012).

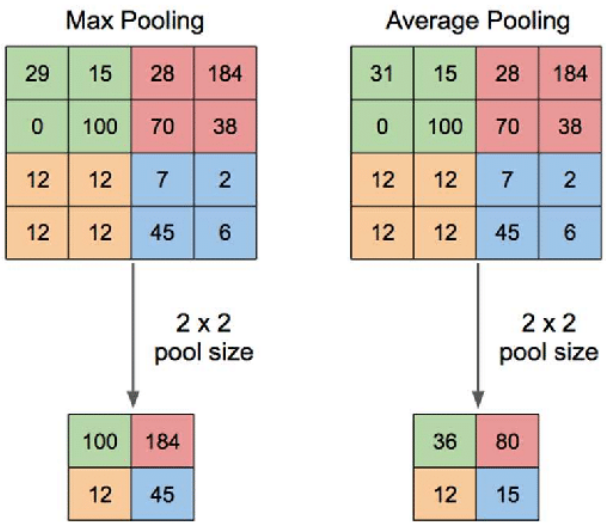


Figure 7: Max and Average (Mean) Pooling

Source: Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail Retrieved from: (Yani, 2019)

4.2.4 Fully Connected Layer

After convolution, clustering and activation, for the final high-level reasoning in the network, a fully connected layer will eventually be used. This layer can be regarded as a node of a regular neural network, weighted and summed the input and added bias. The nodes of this layer are connected to each node of the previous layer. In theory, mapping the visual features extracted by convolution to the desired result is the final learning stage (Schmidhuber, 2015).

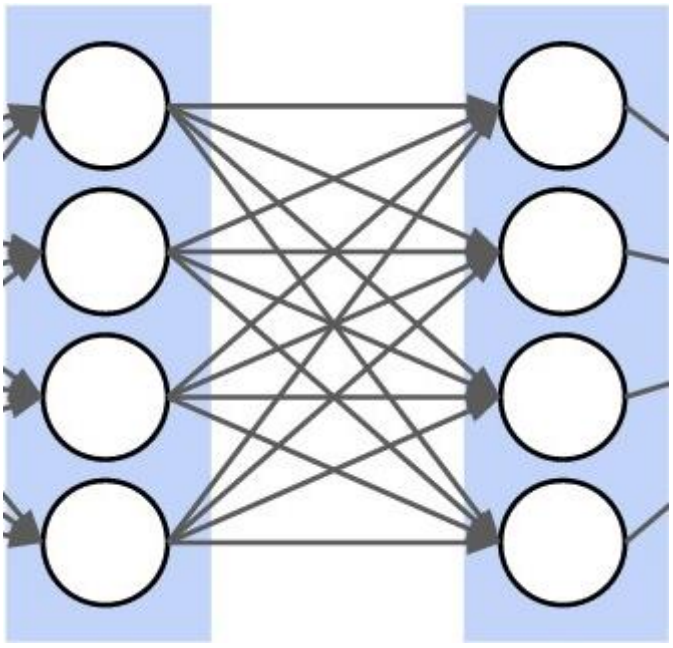


Figure 8: Example of a Fully Connected Layer

4.2.5 Softmax

The Softmax layer is used as the last layer of the classification problem with discrete class labels. This is because it is more appropriate to have an output p_j for each class j. p_j is interpreted as the probability of class j for the input. Softmax assigns a decimal probability that adds up to 1 for each class (Figure x). The output p_j of node j is given by

$$p_j = \frac{e^{z\frac{L}{j}}}{Q}, \quad Q = \sum_{k=1}^{m} e^{z\frac{L}{k}}$$

where
m Is the number of classes
$z\frac{L}{k}$ *is the output of node k of the last layer* (L)

For a given input, the node with the highest probability value is determined as the most probable class.

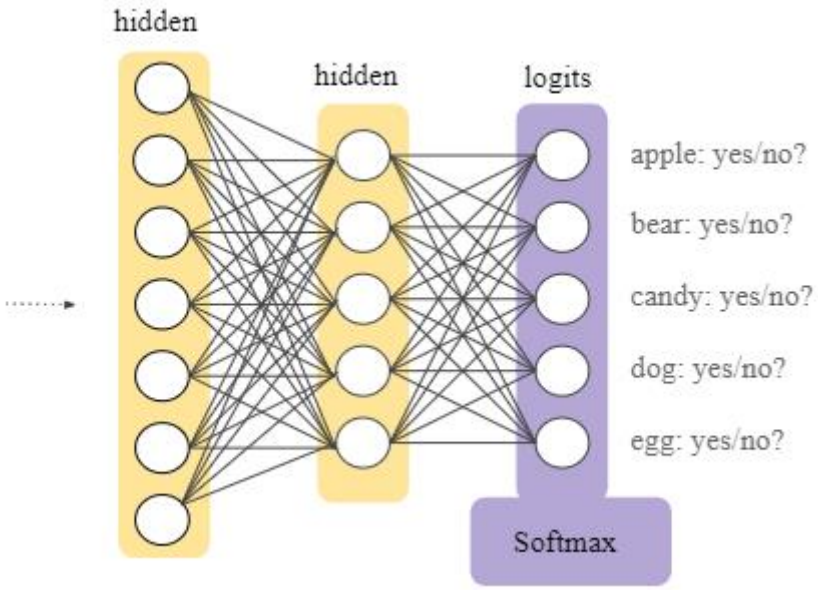


Figure 9: Example of Softmax Layer

Source: Multi-Class Neural Networks: Softmax
Retrieved from: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax

4.3 Loss Function

After the input passes through all the layers of the network, it passes through the softmax layer to get some values in the output layer. The purpose of the loss function is to show how good these values are for the classification task in question. If the value is good, it means that the weights and biases of the first few layers are good. Specifically, the loss function compares the output with the desired result (it knows what the result of the label in the training sample should be).

It measures the degree of inconsistency between the network's predicted value ($\hat{y}$) and the actual label (y). Different types of loss functions can be used, such as root mean square error, L2 loss (Czarnecki, 2017), KL divergence, cross entropy, etc. High losses usually mean that the weights and biases are far apart and require a lot of changes. This information is transmitted to the network through backpropagation.

4.4 Back-Propagation

Backpropagation is an algorithm used to reset the weights in the network according to the result of the loss function. Calculate the gradient of the loss function relative to the weight of the net (the direction of the steepest ascent). Since you need to move in the direction that minimizes the loss function, the weights and biases are reset by moving in the opposite direction of the gradient. This process is called (Stochastic) gradient descent (SGD). The gradient of the last (output) layer is found by finding the partial derivative of the loss function with respect to the weight of the previous layer. In the inner

layer, the local gradients are calculated relative to the output of the next layer, which are called local gradients. From the last layer, calculate the loss. Based on this gradient, the weight of the previous layer is modified, and the local gradient propagates backward to allow the loss of information to flow into the previous layer. This allows an efficient calculation of the gradients in each layer, rather than the naive method of calculating them separately in each layer (al, 1994).

4.5 Regularization

When it comes to deep multi-layered networks, overfitting is a serious problem. This happens when the model is too close to the training data. The problem with this is that the model even tries to fit outliers in the training data, and in the process the decision limit becomes too complicated and moves away from the actual limit (Figure 10). This results in very high training precision, but when new samples are classified according to this limit, they will be misclassified. Therefore, the precision of the test becomes low.

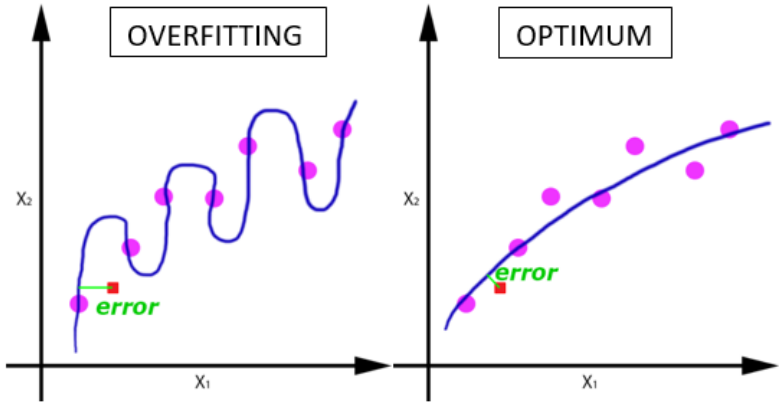


Figure 10: Example of Overfitting

Source: Overfitting vs Underfitting
Received from: https://docs.paperspace.com/machine-learning/wiki/overfitting-vs-underfitting

Dropout is one of the most effective regularization techniques used to solve this problem (N. Srivastava, 2014). In this method, except for the nodes of the output layer, all nodes of the layer behave like a Bernoulli distribution graph with probability p. The probability of each node appearing in the network is $1 - p$. This leads us to use different subnets in each batch, and the final weights and deviations obtained are the average of the weights and deviations obtained from each subnet. In each batch during training, some randomly selected trigger units are deactivated. In other words, their weight and bias are zero. This is only done during the training phase. During the test, the average of the weights obtained by training all subnets is used. In order to implement dropout, the function of each node is replaced with the dropout version (N. Srivastava, 2014).

$$a\frac{l}{j} = \frac{1}{1-p} d\frac{l}{j} \emptyset(z\frac{l}{j})$$

$where$

$d\frac{l}{j} \sim Bernoulli(1-p)$

$a\frac{l}{j}$ $is\ the\ output\ of\ node\ j\ in\ layer\ l$

$z\frac{l}{j}$ $is\ the\ input\ to\ said\ node$

$\emptyset$ $is\ the\ original\ function\ that\ the\ node\ should\ perform$

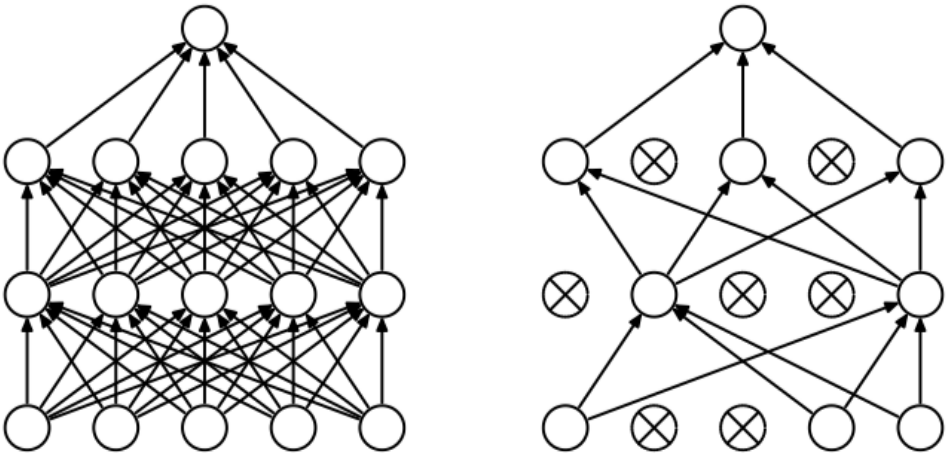


Figure 11: Example of dropout in neural networks

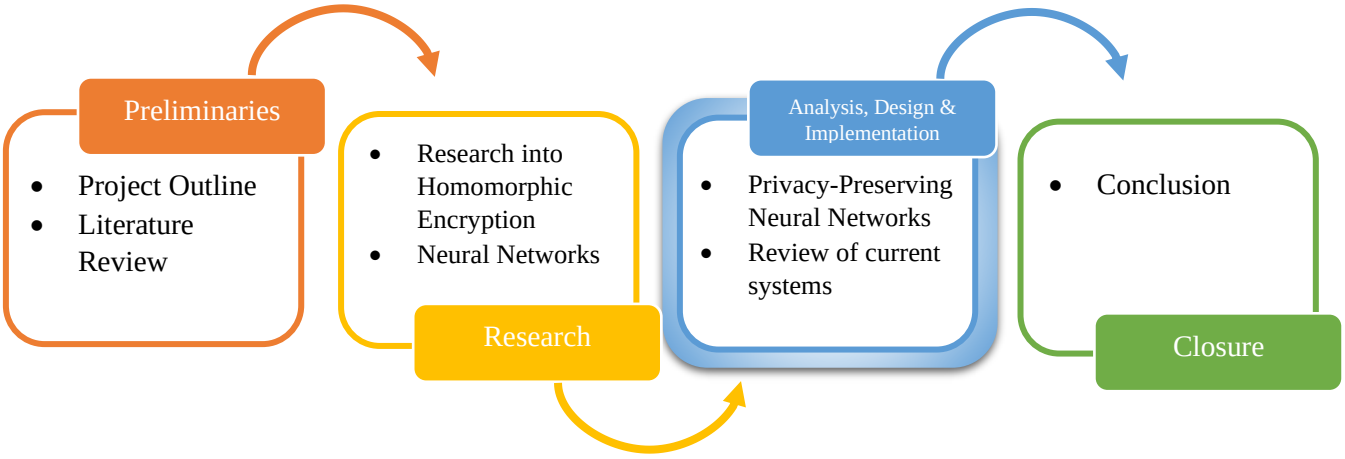
Source: (N. Srivastava, 2014)

Summary of Part Two

Part two consisted of two chapters, chapters 3 and 4. Both chapters focused on looking into the core concepts in respect to Homomorphic Encryption and Neural Networks respectively. In chapter 3 we discuss the development of fully homomorphic encryption, starting by discussing the definition of encryption and expanding through Partially Homomorphic Encryption, Somewhat Homomorphic Encryption and finally review the current and most refined concept of Fully Homomorphic Encryption. In chapter 4 we discuss the theory behind neural networks and focus on the components that important for homomorphic encrypted neural networks. We introduced concepts such as convolutional neural networks and explained the contrasting parameters such as the loss function, back-propagation, and noise/errors. which will prove important in future analysis.

It is important to note that in this section we discussed just the fundamentals of privacy-preserving neural networks and in the following sections we will discuss how they can be used in combination with one and other.

Part Three: Analysis, Design & Implementation



Chapter 5: Privacy-Preserving Neural Networks

5.1 Homomorphic Training of Neural Networks

The training process includes the development of a mapping from the input space to the output space according to the modification of the weight w of each neuron. The network can learn and summarize information based on various examples. The existence of external entities that control the learning process is defined as supervised learning (SL). at the same time. The absence of entities is represented as unsupervised learning (UL).

In supervised learning, when the expected output $\hat{\vartheta}$ is different from the actual output ϑ, the supervisor will adjust w. The loss function $L(\hat{\vartheta}, \vartheta)$ measures the error between the two outputs. In general terms, the training process is as follows:

1) the network receives an input pattern x
2) it calculates the output ϑ = Net(x) which feeds x forward and executes all the calculations until the output layer
3) the network computes $L(\hat{\vartheta}, \vartheta)$
4) modify the weights to reduce the error
5) lastly, the network concentrates on the data and repeats the process several times for all the samples.

The goal is to find a set of weights that minimizes this error. The neural network must provide the correct response after the training process, even if there are patterns that are not used to train the model. In the testing phase, the neural network is evaluated using a set of examples other than the training phase samples. The idea is to avoid overfitting the model and measure its efficiency with new information. The general process is an analogy to the evolution of the human brain in a person's life. The training includes computationally intensive tasks, even for non-homomorphic encryption models. Training the neural network itself involves many operations to find the set of weights that minimizes $L(\hat{\vartheta}, \vartheta)$. With homomorphic encryption, it becomes more challenging even with advanced technology.

In the field of homomorphic encryption, the training process involves a large number of encrypted messages and multiple starts and runs. Its computational complexity is the order of magnitude of unencrypted training. For example, the computational cost of training a seven-layer CNN using a traditional CPU is about 1 hour, while training the same CNN using homomorphic encryption requires about a year (T, 2020). Therefore, it is not practical to train deep neural networks that can contain tens, hundreds, or even thousands of layers.

There are two common options for handling bootstrapping during homomorphic encryption neural network training: acceleration and deletion. High-performance, distributed, parallel computing provides tools for training on large, encrypted data sets. The main trend is to use hardware accelerators such as high-performance computing units (GPUs, FPGAs, etc.) and custom chips (ASICs). These technologies significantly reduce runtime and make them comparable to similar unencrypted versions. The use of cloud computing is the keystone of this direction.

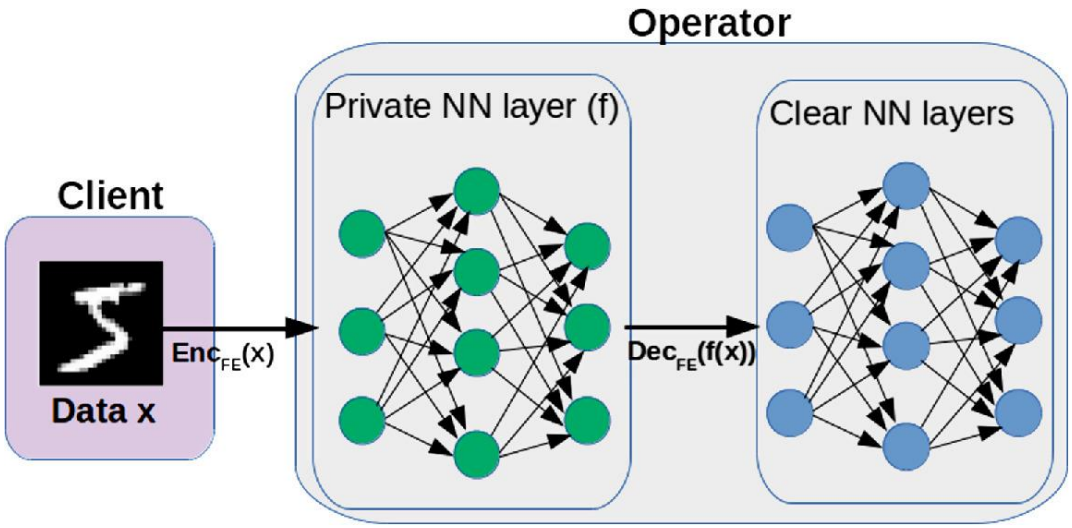


Figure 12: Example of a client passing encrypted data through a neural network

Source: Computing Neural Networks with Homomorphic Encryption and Verifiable Computing
Received from: (Madi A., 2020)

5.2 Homomorphic Evaluation of Neural Networks

The neural network training and testing process does not restrict access to raw data. Therefore, legal and ethical requirements may preclude cloud-based solutions for many applications (Tchernykh A, 2019) (Miranda-Lopez V, 2020). For example, DNA sequencing processing is still a complex task, but thanks to methods developed in the past two decades, it is now faster and cheaper (Kidd JM, 2008). Two opposing considerations must be evaluated. On the one hand, the use of complex models of neural networks to share and process gene sequences can provide a lot of value, such as disease diagnosis. On the other hand, from a privacy perspective, when a person shares their DNA sequence, their siblings or children's DNA is partially shared; if such information reaches an untrusted third party, you can encounter serious trouble. We can find others related to this example through medical, financial or behavioural data.

The above issues have sparked interest in evaluating arbitrarily complex neural networks on encrypted data. A strong direction focuses on the benefits of homomorphic encryption schemes for neural network information processing. The homomorphic encrypted neural network is a natural extension of the conventional neural network model. The method includes applying homomorphic encryption to the network input and propagating the signal through the network in a homomorphic manner. The training and inference phases are the foundation of the neural network process to protect privacy. In the inference stage, the dimensions of the model are known in advance, so the number of operations can be estimated a priori.

From the perspective of homomorphic encryption, the network represents a levelled circuit, where the levels are called layers. The cryptographic system allows the use of a predefined noise budget to implement an encryption scheme, avoiding booting or any re-encryption functions. In other words, since the amount of noise supported by the ciphertext is known, and the polynomial function has a fixed maximum degree in the encrypted data, the level homomorphic cipher scheme is sufficient for the inference stage.

However, deep learning requires an invariable fully homomorphic encryption scale scheme because it involves many hidden layers in the network. Therefore, a lot of noise must be dealt with by bootstrapping or any re-encryption function. In addition, since the homomorphic cryptographic scheme only supports addition and multiplication, only polynomial functions can be calculated. The construction of a homomorphic encrypted neural network involves two challenges: the computational

design of homomorphic processing of internal network functions and the operation of low-order polynomials. The following sections delve into the challenges and solutions presented in the literature.

5.4 Data Manipulation

The representation and operation of data in the network is another challenge in the construction and implementation of homomorphic encryption neural networks. Each ciphertext c has some noise that hides the message. Therefore, $c = m + e$, where m represents the original message and e defines noise. Regardless of the nature of the data sent in these computational models, their passwords are all entries in a polynomial ring. Assigning whole numbers, floating point numbers, binary numbers, or other number systems to polynomials is a standard and widely studied exercise. For example, traditional integer mapping techniques use their binary representation; binary integer expansion generates the coefficients of the polynomial. However, it is still necessary to implement the low-order polynomial representation.

Many methods provide low-order polynomial representations by applying the Chine Remainder Theorem (CRT) (Khedr A, 2015) (Dowlin N, 2016), where k primes $q_1, \dots, q_k$ and a polynomial $\sum a_i x^i$ define k polynomials such that the j^{th} polynomial is $\sum[(a_i) mod t_j] x^i$. CryptoNets (Dowlin N, 2016) and AlexNet (Badawi A Al, 2018) follow the CRT method with a high degree of polynomials. They implement the Single Instruction Multiple Data (SIMD) technology in which a single polynomial encodes the characteristics of multiple instances. For example, if the input data is an image with a size of 28 × 28 pixels, it will generate 784 ciphertext (one for each pixel), where the number of images determines the size of the ciphertext. This construct uses CRT to break down the ciphertext into multiple blocks and process them in parallel. The main contribution of CRT is to reduce the processing time of high-order polynomials.

The single statement multiple data method has some latency and memory limitations. A single prediction means many operations, because each characteristic represents one message, and a large number of messages can create a memory bottleneck. It makes its implementation in deep neural network models unfeasible. The biggest disadvantage is the low efficiency. The computing resources consumed to process one or a thousand images are the same.

LoLa (Brutzkus A, 2019) uses alternative representations to reduce the latency and memory usage of its predecessors. Encode the d-dimensional input vector v as a single message m, where $m_i = v_i$, because $i = 1, \dots, d$. Compared with the d message and $O(d)$ operation of the same classifier in the previous system, the private prediction of the linear classifier on v requires only a single message and $O(logd)$ operation. However, the pattern still depends on the dimensions of the message, so it is not practical for real applications with big data.

In short, the contributions of CryptoNets and LoLa are the foundation of many types of solutions. Later work combined different techniques to reduce and manipulate polynomials, such as CRT and SIMD, among other methods. Furthermore, the evolution to the NNHE-compatible model suggests a combination of GPU, Field Programmable Gate Array (FPGA), and Application Specific Integrated Circuit (ASIC). We are facing an emerging field of research with myriad potential and quantifiable benefits.

5.5 Noise and Errors

As shown in the previous sections, homomorphic encryption sometimes uses the concepts of noise and error as interchangeable concepts. In this section, we distinguish them from each other for a better understanding. Generally speaking, noise is information that injects data into the overall structure of a public key cryptosystem. Instead, the error appears as part of the rounding error that occurs during the approximation calculation. This section introduces the concepts of noise and error in the context of machine learning, especially in neural network models.

4.6.1 Noise

The design of a homomorphic encrypted neural network must consider the number of arithmetic operations (addition and multiplication) required for its realization. In the BFV (Fan J, 2012) and CKKS (Cheon JH, 2020) schemes, the number of additions is unlimited, but the number of multiplications is limited. The multiplication depth is the maximum number of homomorphic multiplications, which can be performed in ciphertext to correctly retrieve the results of these multiplications. The computational complexity of each multiplication and the size of the public key depend on the depth of the multiplication. Both of these parameters increase with increasing depth. Practice shows that the amount of noise depends on two factors: level key and realization.

- Level keys. The upper limit of the multiplication depth depends on the selected level key. For example, the PALISADE library provides an algorithm for selecting level keys to change the amount of noise that cannot be eliminated without decryption or bootstrapping.
- Implementation. The maximum number of multiplication depths also depends on the algorithm implementation.

Gentry (C, 2009) proposed a mechanism to eliminate redundant noise through the start-up process (see Section 3.2), which can be broadly interpreted as re-encryption or decryption in the form of encryption. Some noise can only be eliminated by decryption. For integer homomorphic encryption, the amount of noise after the start-up process is greater than the amount of noise in the original ciphertext.

The general neural network model consists of convolution and activation mathematical functions. At best, the multiplication depth of the mathematical convolution is equal to 1. For the activation implementation, the function is approximated by a polynomial, because the homomorphic cipher only supports addition and multiplication. In the worst case, the depth of the multiplication is equal to the binary logarithm of the polynomial plus one. To reduce the depth of multiplication in polynomial calculations, various techniques based on combinations of functions are used.

4.6.2 Errors

Errors in data processing will affect the accuracy of the results. The realization of homomorphic encryption neural network distinguishes two types of errors: algorithm errors and execution errors.

Algorithmic error. Using integer polynomials to represent real numbers can lead to incorrect results. Even if noise is not added, errors may occur in numbers close to zero. Converting the value vector to a polynomial $m(X)$ will cause serious errors in the calculation. Therefore, when the input data is normalized, that is, the value is compressed to the interval $[0, 1]$, the result of using neural network homomorphic encryption is likely to be wrong. In addition, when using unstable algorithms, this type of error can lead to incorrect results.

Running error. Due to the polynomial approximation of the activation function of the neural network, calculation errors can appear. For example, consider the function $ReLU(x) = max(x, 0) = (sgn(x) + 1) \cdot x$. The function $max(\boldsymbol{a}, b)$ returns the maximum value of a and b, where $sgn(\boldsymbol{x})$ is a sign function. When we calculate $ReLU(\boldsymbol{x})$, for $x < \mathbf{0}$, by homomorphic encryption, using the sgn (x) algorithm (Cheon JH, 2020), the function will be greater than zero. Therefore, the implementation of homomorphic encryption neural networks must consider possible rounding errors.

Chapter 6:
Review of Current Systems

6.1 Neural Network – Homomorphic Encryption Tools

In this section, we describe some privacy-preserving neural network implementations in real-world applications and show the current tools developed by them. First, we explain the progress and limitations of the homomorphic cipher library and focus on the neural network framework.

Product	Creator	Language	License
SEAL	Microsoft	C++	MIT
HElib	IBM	C++	Apache-2.0
TFHE	Gama et al.	C++	Apache-2.0
HEAAN	CryptoLab. Inc.	C++	CC-BY-NC-3.0
PALISADE	New Jersey Institute of Technology	C++	BSD-2-Clause
Cingulata	CEA LIST	C++	CECILL-1.0
FV-NFLlib	CryptoExperts	C++	GPL-3.0-only
Lattigo	Laboratory for Data Security	Go	Apache-2.0

Figure 13: A table of popular homomorphic encryption tools

High-quality implementation should complement theoretical research. In recent years, industrial and academic groups have launched several HE libraries: SEAL, HElib, TFHE, PALISADE, etc. Many implementations are based on errored ring learning and include general underlying ring selection, error distribution, and other parameters.

The Simple Encrypted Arithmetic Library (SEAL) (Chen H, 2017) is the most widely used open-source homomorphic encryption tool, supporting BFV and CKKS schemes. It is implemented in C ++, and other languages C #, F #, Python, and JavaScript have been actively developed. SEAL can compress data to achieve significant savings in memory footprint.

HomomorphicEncryption Library (HElib) (Halevi S, 2013) is an open-source library based on the BGV schema and developed in C ++. It focuses on the efficient use of ciphertext packaging and optimizing data movement. One downside to the HElib is its limited boot performance.

The Fastest Fully Homomorphic Encryption (TFHE) (Brakerski Z, 2011) is an open-source library that supports ring variants and alternative representations of torus of GSW. The C/C++ library implements very fast door-to-door startup procedures; this mode does not limit the number or composition of doors.

PALISADE (PALISADE, n.d.) is an open-source project for implementing HE libraries that support BGV, BFV, CKKS, FHEW, and THEW schemas. It is developed in C ++ and provides extensions for many parts. PALISADE uses the residual number system algorithm to achieve high performance.

Choosing a specific homomorphic cipher library is a step that must understand its advantages and disadvantages. Figure 14 compares the most common general homomorphic cipher libraries and their advantages and disadvantages. Describes the key characteristics of rapid adoption and further development. These basic characteristics should be taken into account when selecting the appropriate privacy protection tools and methods.

Tool	Pros	Cons
SEAL	Well-documented Easy security parameters setting	Poor flexibility Limited number of supported schemes
HElib	Efficient homomorphic operations	Low bootstrapping performance Complicated security parameter setting
TFHE	Fast bootstrapping	Poor performance for simple tasks
PALISADE	Multiple homomorphic encryption schemes Cross-platform	Poor documentation and support

Figure 14: A table comparison of four different popular encryption tools

6.2 Neural Network – Homomorphic Encryption Implementations

This section introduces the comparison of three homomorphic encryption neural network implementations. We will compare the performance of high-level and low-level tools. In addition, we will demonstrate the promise between technologies and highlight their potential combinations. Most homomorphic encryption tools are developed in low-level languages such as C or C++, and the idea is to increase the speed of applications. The main development tools of neural network are implemented in high-level languages such as Python, which is convenient for quick understanding. Neural network frameworks greatly simplify new development methods, but their structure is not designed to support alternative information representations.

The goals of a homomorphic encrypted neural network system are three: accuracy, data security, and computational complexity. If extremely safe models cannot provide acceptable calculation accuracy and complexity, they are useless. Therefore, all three goals should always be reflected in the system design. This is where the trade-offs between technologies come into play.

For learning purposes, we use PyTorch (Paszke A, 2019) as a representative machine learning tool. It is the deep learning framework used in many articles from major research conferences in recent years. PyTorch is also popular in the research community and industry. A friendly and flexible environment is your main goal. Users can quickly and easily conduct experiments on the platform.

The effective implementation of homomorphic encryption neural networks requires the flexibility of homomorphic and neural network encryption technology, combining the potential of homomorphic encryption tools with PyTorch facilities. The PySyft, CrypTen, and TenSEAL libraries provide a bridge between the PyTorch platform and many privacy protection technologies described in the long history of academic research. We compared the performance of three neural networks implemented in PyTorch, SEAL and a combination of the two (SEAL + PyTorch). SEAL is a low-level tool, while PyTorch is a high-level tool. Both have high adoption rates in their respective fields. The combination of the two methods provides a compromise between efficiency and ease of implementation.

The evaluation considers three types of instances: native, small, and large. The native instance (N) corresponds to the original implementation using the specifications, parameters and features proposed in the corresponding document (Dowlin N, 2016). The small instance (S) has only one convolutional layer. The large example (L) uses a convolutional network with a six-layer architecture: a convolutional input layer with 28 × 28 input nodes, four hidden layers, and a fully connected output layer with ten neurons (one for each number or category). The difference between a small instance and a large instance is the number of convolutional layers. The four hidden layers include a clustering activation layer, a convolutional layer, a grouping layer, and a fully connected activation layer. All neurons use a square activation function.

We used two data sets, including the famous MNIST (Modified National Institute of Standards and Technology) handwritten digit database and the MNIST-like fashion data set (FMNIST) for labelling

fashion images. They are commonly used to train various image processing systems. The test set includes 10,000 examples.

Figure 15 shows the accuracy, latency, and memory usage of the three convolutional neural networks with MNIST and FMNIST. Latency corresponds to the time required to process a single prediction request. CryptoNets has the same processing time for 1 or 4096 predictions. The results show the trade-offs between the methods implemented in memory and latency. Compared with other methods in languages such as C++, the computing resources of the framework implemented in Python are compromised.

Tool	HE	NN-HE	Instance	MNIST			F-MNIST		
				Lat. (s)	Acc. (%)	Memory (Mb)	Lat. (S)	Acc. (%)	Memory (Mb)
PyTorch			L	0.011	99.51	465.20	0.010	89.20	448.42
SEAL	*	CryptoNets	N	283.56	98.95	4750.21	290.01	83.74	5781.14
	*	LoLa	S	0.34	96.92	593.70	0.24	82.02	576.32
	*	LoLa	N	3.50	98.95	2076.40	3.45	83.74	2010.90
	*	LoLa	L	14.43	99.20	2781.60	14.21	84.13	2430.10
SEAL + PyTorch	*	-	S	1.2	96.90	1543.34	1.16	82.01	1453.21
	*	-	N	12.67	98.88	7301.92	12.52	83.71	7135.67
	*	-	L	46.92	99.16	9871.81	46.11	84.08	8869.10

Figure 15: A performance comparison of three different neural networks on encrypted data

We are at the beginning of a long way to explore the field of privacy protection neural networks. Due to higher computational requirements and complex functional space, several methods are limited. However, this analysis clarifies the competitive precision performance of the PyTorch + SEAL implementation.

6.3 Challenges

Although the homomorphic encryption standards, platforms, and implementations presented in this report help advance the development of privacy-preserving neural networks, there are still some open challenges that need to be resolved: overhead, performance, interoperability, bootstrapping bottlenecks, sign determination, etc

Overhead

Compared to their unencrypted analogous, homomorphic encrypted neural networks have significant overhead, which makes them impractical in many applications. The neural network training phase includes the computationally intensive tasks of the non-homomorphic encryption model. Using homomorphic encryption, even with today's technology and computing power, it becomes more challenging. A new trend is to avoid the training phase by using previously trained models to strike a balance between complexity and accuracy.

Parallelization

One way to deal with computational overhead is to combine new and familiar parallelization techniques. The homomorphic encryption neural network model can be adapted to use high-performance computing, distributed systems, and dedicated resources. Multi-core processing unit (GPU, FPGA, etc.) or custom chip (ASIC) technology brings the possibility of a friendlier and more efficient homomorphic encryption neural network environment. Another way to improve overall efficiency is related to the ability of batching and parallelize multiple bootstrapping operations together.

Polynomial Approximation

A key challenge in the development of homomorphic encrypted neural networks includes the computational design of the homomorphic processing of the internal functions of neurons. The homomorphic encryption neural network requires operations that are not compatible with

homomorphic encryption, so it is necessary to find an alternative password-compatible function to operate with encrypted data. The activation function is the building block for building an efficient homomorphic encryption neural network. Determines the accuracy and computational efficiency of the network. In addition, the activation function has a significant impact on the speed of network convergence. Furthermore, its derivative, also called the gradient, is the basis of the training phase. A variety of methods solve the restriction by using a polynomial form with the same password to approximate incompatible functions by polynomial. These functions must exhibit a balance between complexity and precision, which limits the efficiency of traditional approximation techniques (Cortés-Mendoza JM, 2020).
In practice, insufficient approximation function will lead to poor homomorphic encryption neural network performance and excessive processing time. Also, it generates larger encrypted messages, which increases memory usage. The challenge of designing a password so that the activation function is approximate is to identify low-order polynomials with minimal errors and good precision.

Levelled Homomorphic Encryption Schemes
Another important direction is focused on designing solutions that do not require guidance to support the neural network assessment of limited depth (default). This hierarchical homomorphic encryption scheme significantly improves performance by removing the bottleneck and complexity caused by the heavy boot encryption feature. However, this method limits the implementation of deep learning. Although effective for narrow neural networks with homomorphic encryption, the complexity can be very high for deep learning models.

Binary Neural Networks
Binary neural networks have become an opportunity field for realizing non-interactive blind homomorphic encrypted neural network models. Since the function space is limited, the solution must be limited by the number of possible inputs and outputs. In a binary neural network, each layer uses a set of binary weights and a binary activation function to map a binary input to a binary output. For deviations, batch normalization is applied before each trigger function. The input data is binarized using predefined thresholds.
Generally speaking, the data and weights in the non-standard binary representation {-1, 1} can be allocated to the binary space {0, 1} by replacing -1 with 0. Weighted summation can be performed by multiplying the elements. Using the logical operator XNOR, and then add the results of the previous step by calculating the number of ones. If y> 0, the binary activation function f (y) returns 1, otherwise it returns -1.

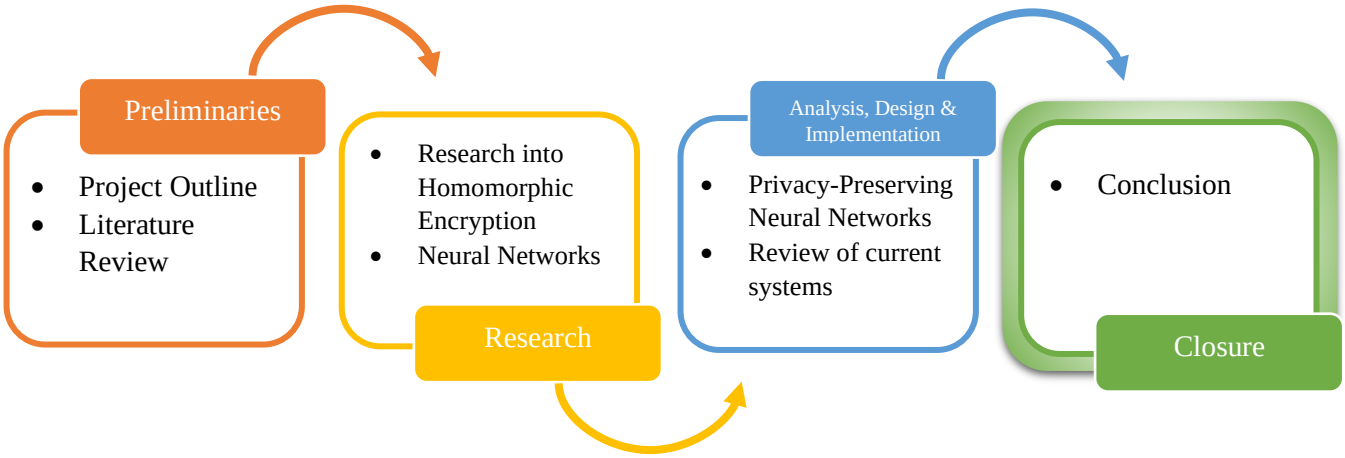
Interoperability
The interoperability of existing machine learning tools is another challenging problem in modelling friendly neural networks of homomorphic encryption. Popular neural network frameworks simplify the development of new neural network methods, but do not provide homomorphic encryption support. The development of homomorphic encrypted neural networks relies on current tools and their flexibility to provide or incorporate new methods. The low flexibility of various homomorphic cipher libraries limits their interaction with other frameworks. It makes the design, testing, and deployment of new models more complex, increasing development time.

Automatization
The development of homomorphic encryption applications involves manual configuration and rich experience in different fields: specific mode optimization, complex configuration of security parameters, low-level programming, etc. Incorrect settings can lead to poor performance, insecure encryption, and information corruption or irrecoverability. It needs to automate and simplify the development life cycle. The implementation should be easy to use by beginners and highly configurable for expert users.

Summary of Part Three

The third part consists of two chapters, Chapter 5, and Chapter 6. Chapter 5 includes methods for training and evaluating neural networks on fully homomorphic encrypted data. First, we discuss the process behind the user / client passing encrypted data through the neural network. This includes calculating the initial layer and performing all the calculations down to the last layer. Also, modify the weights to reduce system errors and discuss how to use the boot loader to encrypt data. Next, we will see how to use data manipulation, noise, and errors to evaluate neural network performance. In Chapter 6, we discussed some real-world encryption tools and homomorphic encryption neural network applications. Here, we briefly evaluated the performance of three different neural networks implemented using PyTorch and SEAL. Finally, we reviewed some potential challenges and pitfalls related to privacy-preserving neural networks.

Part Four: Closure



Chapter 7: Conclusion

7.1 Evaluation of Overall Work

In this section, we will evaluate the general work that has been done on this project. We will discuss our achievements, the difficulties we encountered in this project, and the many problems we face. We will evaluate our work against the goals we defined in the introductory chapter and propose what can be done in future work to further improve this project.

Overall, I personally think I have done a good job on this project. Originally, when first theorising this report I set out to develop my own encryption and neural network system. Although it wasn't until, I had done thorough research that I realised that the coding was far too arduous for me to tackle (there still isn't a complete concept that fully solves the issue of privacy preserving neural networks). This in combination of choosing a topic I have not worked on before, not even any basic introductory knowledge caused a significant learning curve that I was not fully able to overcome. In addition, despite not having the slightest bit of help in the development of research, I believe that the work that I have done here is very advanced. Overall, I am very happy that I managed to understand the complicated mathematics behind homomorphic encryption and was able to add it on top of my existing knowledge of neural networks. This may be a new starting point for my further research in the field of cryptography and machine learning.

7.2 Work Achieved

As mentioned in a previous section, the concept of privacy preserving neural networks and privacy preserving machine learning is an on-going concept within the real-world. Therefore, it is difficult to assign a goal to my work that was to fully functional solution within the time given. Therefore, it was my goal to thoroughly research all available concepts and possible research potential improvements for the future by analysing the short comings of fully homomorphic encryption.

I think I have achieved success in researching and comparing basic neural networks that protect existing privacy. I have studied many technologies and even explored the mathematical concepts behind them. In addition, I was able to organize all the necessary information to continue such research in the future, which is valuable for anyone who wants to undertake similar projects. As it can be seen I have an extensive list of references which should prove the scope of my research within this project.

In the end, I did a lot of work that I did not reflect in this report. In order to develop some of the concepts in this report, I had to experiment with very basic open-source systems. Maybe these are very limited and do not help the project, so they have to be scrapped. In addition, I was able to have a basic understanding of C++, which I had never encountered before. It took several hours of training to get used to it, and understanding the open source code used to perform homomorphic encryption proved very difficult.

7.3 Evaluation against the objectives

At the beginning of this report, we defined a number of objectives to be carried out during this project in order to consider this project "successful". In this section we will assess each of the objects against the work achieved.

1. **Explore and understand some of the various elements within in the project such as Neural Networks, Encrypted data or more specifically homomorphic encrypted data.**

As required, I have completely looked into various methods of privacy preserving neural network. All concepts and techniques are well documented and explained.

The techniques I have researched cover several complex systems such as loss function, back-propagation, regularization, noise and errors. Furthermore, we discussed the implementation not only fully homomorphic encryption but also Partially Homomorphic Encryption and Somewhat Homomorphic Encryption. We also explored the timeline associated with the development of fully

homomorphic encryption. If I was to redo the project once again, I would like to explore some further concepts of MLaaS. That includes multiple forms of machine learning tools not only neural networks. I believe this could have provided some interesting literature when comparing the different potential benefits and limitations of differing machine learning techniques.

2. **Review and compare a few of the current solutions/systems**

In chapter 6 I was able to briefly compare some current solutions/systems in the form of PyTorch + SEAL. These were compared over three different instances: native, small and large. I was able to produce an interesting table with some relevant results but personally I was disappointed. Originally, I aimed to produce a comprehensive and detailed analysis of 5+ current systems but was unable to complete this due to the unforeseen coding experience needed. Even though I feel like this objective was not fully met I believe I was able to at least produce some basic analysis of current systems and report on some of the results found.

3. **Develop a Privacy-Preserving Neural Network system**

This task was completely failed as it proved far too difficult for me to develop any meaningful advancement within the time given. I believe this goal was far too optimistic for a topic that is already so modern, and I also vastly overestimated my own coding abilities in addition to learning a whole knew programming language. If I was to run this project again, I would have chosen a goal more suitable to my abilities in more high level programming languages such as python and R and also focused on machine learning and data analytics instead of cryptography.

7.4 Reflections and Potential Future Work

In this section, I will discuss my personal thoughts on the project and suggest what can be done to improve in the future.

I believe when starting this project, I did not fully understand the complexity of homomorphic encryption. Originally when I started the project, I wanted to develop my own privacy-preserving neural network. However, after several days of attempts I concluded that the goal was far above the capacity of a MSc dissertation. As a result, I decided to solely focus on providing a comprehensive and detailed research instead.

In the end, I think I did some useful work and benefited from expanding my understanding of other applications of machine learning in the real world. In particular, for areas such as finance, which I am personally interested in, the privacy-protecting neural network seems to have become a very important concept. Therefore, I believe that the concepts mentioned in this project will be very useful in the future.

7.5 Conclusion

The easy implementation of machine learning in the cloud makes homomorphic encryption an important mechanism for solving security and privacy issues. Many solutions have been proposed on the continuum of theory and application. Theoretical research increases understanding of the problem by developing new theories and algorithms, while applied research is for solving practical problems in business, medicine, industry, and other fields. Several aspects of privacy preserving neural networks have been discussed in this report, but a methodical comparison of the state-of-art homomorphic encrypted solutions has not yet been created.

In this paper, we:

- Review the latest advances of Homomorphic Encryption cryptosystems, focusing mainly on the intersection of cryptography and neural networks.
- Describe basic concepts that are easy for readers who are not familiar with privacy protection neural networks, such as bootstrapping, key-switching, noise, running errors, algorithm errors,

etc. We cover the main theoretical results, capabilities, opportunities, potential applications, and design and application trends.

- Highlight the trade-off between neural network technology and homomorphic feasibility, combining their potential and design and application trends.
- Discuss current development tools, frameworks, emerging trends in research, and relevant application domains.
- Compare three privacy-preserving neural networks with CryptoNets and LoLa, using high- and low-level tools to show the advantages and disadvantages of each method.
- Finally, we discuss open problems, challenges, and solutions related to homomorphic cryptography and machine learning.

References

A. Krizhevsky, I. S. a. G. E. H., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems,* p. 1097–1105.

al, L. V. F. e., 1994. Fundamentals of neural networks: architectures, algorithms, and applications. *Prentice-Hall Englewood Cliffs,* Volume 3.

Alperin-Sheriff J, P. C., 2014. Faster bootstrapping with polynomial error. *Advances in Cryptology – CRYPTO,* 8616(https://doi.org/10.1007/978-3-662-44371-2_17), p. 297–314.

Armknecht F, B. C. C. C. G. K. J. A. R. C. S. M., 2015. A guide to fully homomorphic encryption. *IACR Cryptology ePrint Archive,* Volume 1192.

Badawi A Al, C. J. L. J. M. C. S. J. T. B. N. X. A. K. C. V., 2018. Towards the AlexNet moment for homomorphic encryption: HCNN, the First Homomorphic CNN on Encrypted Data with GPUs. *IEEE Transactions on Emerging Topics in Computing,* Issue https://doi.org/10.1109/TETC.2020.3014636.

Baskin, C. &. L. N. &. M. A. &. Z. E., 2017. Streaming Architecture for Large-Scale Quantized Neural Networks on an FPGA-Based Dataflow Platform.

Boneh D, G. E. N. K., 2005. Evaluating 2-DNF formulas on ciphertexts. *Theory of cryptography,* 3378(https://doi.org/10.1007/978-3-540-30576-7_18), p. 325–341.

Brakerski Z, G. C. V. V., 2012. (Leveled) fully homomorphic encryption without bootstrapping. *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference,* Issue https://doi.org/10.1145/2090236.2090262, p. 309–325.

Brakerski Z, V. V., 2011. Efficient fully homomorphic encryption from (Standard). *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science,* Issue https://doi.org/10.1109/FOCS.2011.12, p. 97–106.

Brutzkus A, E. O. G.-B. R., 2019. Low latency privacy preserving inference. *Proceedings of the 36th Peer-to-Peer Netw. Appl. (2021),* 14(JMLR.org), pp. 1666–1691, 1689.

C. Orlandi, A. P. a. M. B., 2007. Oblivious neural network computing via homomorphic encryption. *EURASIP Journal on Information Security,* Volume 1, pp. 037, 343.

C, G., 2009. *A fully Homomorphic encryption scheme.,* Stanford University: Stanford.

C, G., 2010. Computing arbitrary functions of encrypted data. *Commun ACM,* 53(https://doi.org/10.1145/1666420.1666444), p. 97–105.

Chen H, L. K. P. R., 2017. Simple encrypted arithmetic library - SEAL v2.1. *Brenner M et al (eds) Financial Cryptography and Data Security,* 10323(https://doi.org/10.1007/978-3-319-70278-0_1), p. 3–18.

Cheon JH, K. A. K. M. S. Y., 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. *Takagi T, Peyrin T (eds) Advances in Cryptology – ASIACRYPT 2017,* 10624(https://doi.org/10.1007/978-3-319-70694-8_15), p. 409–437.

Cheon JH, K. D. K. D., 2020. Efficient Homomorphic comparison methods with optimal complexity. *Moriai S, Wang H (eds) Advances in Cryptology - ASIACRYPT 2020,* 12492(https://doi.org/10.1007/978-3-030-64834-3_8), p. 221–256.

Cortés-Mendoza JM, T. A. B. M. P.-G. L. R. G. L. F. W. X. A. A., 2020. Privacy-preserving logistic regression as a cloud service based on residue number system. *Communications in Computer and Information Science,* 1331(https://doi.org/10.1007/978-3-030-64616-5_51), p. 598–610.

Czarnecki, K. J. a. W. M., 2017. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659.*

Damgård I, J. M., 2001. A Generalisation, a simplification and some applications of paillier's probabilistic public-key system. *Public Key Cryptography,* 1992(https://doi.org/10.1007/3-540-44586-2_9), p. 119–136.

Didie W, H. M., 1976. New directions in cryptography. *IEEE Trans Inf Theory,* Volume 22, p. 472–492.

Diffie, W. H. M., 1976. New directions in cryptography. *IEEE Transactions on Information Theory,* Volume IT-22, p. 472–492.

Dowlin N, G.-B. R. L. K. L. K. N. M. W. J., 2016. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. *Balcan M-F, Weinberger KQ (eds) Proceedings of the 33rd International Conference on Machine Learning (ICML'16),* Issue JMLR.org, p. 201–210.

E.Witchel, T. a. C. S. a. R. a. V. S. a., 2018. Chrion: Privacy-preserving Machine Learning as a Service. *ArXiv,* Volume abs/1803.05961.

Erodotou, M., n.d. *Automated detection and tracking of Mycobacterium Tuberculosis cells,* s.l.: University of Surrey.

Fan J, V. F., 2012. Somewhat practical fully Homomorphic encryption. *IACR Cryptol,* Issue ePrint Arch:2012/144.

Florian Bourse, M. M. M. M. a. P. P., 2017. Fast Homomorphic Evaluation of Deep Discretized Neural Networks. Issue Orange Labs, Applied Crypto Group, CryptoExperts, Paris-France.

Gentry C, H. S., 2011. Implementing gentry's fully homomorphic encryption scheme.. *Paterson KG (ed) Advances in Cryptology – EUROCRYPT,* 6632(https://doi.org/10.1007/978-3-642-20465-4_9), p. 129–148.

Gentry C, S. A. W. B., 2013. Homomorphic encryption from learning with errors: conceptually-simpler, asymptoticallyfaster, attribute-based. *Advances in Cryptology – CRYPTO,* 8042(https://doi.org/10.1007/978-3-642-40041-4_5), p. 75–92.

Goldreich, O., 1998. Secure multi-party computation. Volume 78.

Goldwasser S, M. S., 1982. Probabilistic encryption and how to play mental poker keeping secret all partial information. *Proceedings of the fourteenth annual ACM symposium on Theory of computing,* Issue https://doi.org/10.1145/800070.802212, p. 365–377.

Halevi S, S. V., 2013. Design and implementation of a Homomorphic-encryption library. *IBM Res,* Volume 6, p. 12–15.

Hiromasa R, A. M. O. T., 2015. Packing messages and optimizing bootstrapping in GSW-FHE. *PublicKey Cryptography - PKC,* 9020(https://doi.org/10.1007/978-3-662-46447-2_31), p. 699–715.

J, B., 1994. Dense probabilistic encryption. *Proceedings of the workshop on selected areas of cryptography,* p. 120–128.

Kawachi A, T. K. X. K., 2007. Multi-bit cryptosystems based on lattice problems. *Okamoto T, Wang X,* 4450(https://doi.org/10.1007/978-3-540-71677-8_21), p. 315–329.

K, G., 2004. Subgroup membership problems and public key cryptosystem. *Norwegian University of Science and Technology. PhD Thesis.*

Khedr A, G. G. M. S. V. V., 2015. SHIELD: Scalable Homomorphic implementation of encrypted data-classifiers. *IEEE Trans Comput,* 65(https://doi.org/10.1109/TC.2015.2500576), p. 2848–2858.

Kidd JM, C. G. D. W. H. H. S. N. G. T. H. N. T. B. A. C. A. F. H. E. Z. T., 2008. Mapping and sequencing of structural variation from eight human genomes. *Nature,* 453(https://doi.org/10.1038/nature06862), p. 56–64.

M. Barni, C. O. a. A. P., 2006. A privacy-preserving protocol for neuralnetwork-based computation. *Proceedings of the 8th workshop on Multimedia and security,* p. 146–151.

Madi A., S. R. S. O., 2020. Computing Neural Networks with Homomorphic Encryption and Verifiable Computing. *Applied Cryptography and Network Security Workshops,* 12418(https://doi.org/10.1007/978-3-030-61638-0_17).

Miranda-Lopez V, T. A. B. M. A. A. T. V. D. A., 2020. 2Lbp-RRNS: Two-levels RRNS with backpropagation for increased reliability and privacy-preserving of secure multi-clouds data storage. *IEEE Access. Multidiscip. Open Access J.,* 1(https://doi.org/10.1109/ACCESS.2020.3032655).

M, M., 2018. Fully homomorphic encryption for machine learning. *PSL Research University, PhD Thesis.*

N. Srivastava, G. H. A. K. I. S. a. R. S., 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research,* Volume 15, p. 1929–1958.

Naccache D, S. J., 1998. A new public key cryptosystem based on higher residues. *Proceedings of the 5th ACM conference on Computer and communications security,* Issue https://doi.org/10.1145/288090.288106, p. 59–66.

Okamoto T, U. S., 1998. A new public-key cryptosystem as secure as factoring. *Advances in Cryptology — EUROCRYPT,* 1403(https://doi.org/10.1007/BFb0054135), p. 308–318.

PALISADE, n.d. [Online]
Available at: https://palisade-crypto.org/community

Paszke A, G. S. M. F. L. A. B. J. C. G. K. T., 2019. PyTorch: An Imperative Style, HighPerformance Deep Learning Library. *Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada,* Volume 1912, p. 8024–8035.

P, P., 1999. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology - EUROCRYPT,* 1592(https://doi.org/10.1007/3-540-48910-X_16), p. 223–238.

Qaisar Ahmad Al Badawi A, P. Y. A. K. V. B. R. K., 2019. Implementation and performance evaluation of RNS variants of the BFV Homomorphic encryption scheme. *IEEE Trans Emerg Top Comput.,* Issue https://doi.org/10.1109/TETC.2019.2902799.

R. S. Sutton, A. G. B., 1998. Reinforcement learning: An introduction.

Rivest RL, D. M. A. L., 1978. On data banks and privacy homomorphisms. Volume 4, p. 160–179.

Rivest R, S. A. A. L., 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun ACM,* 21(https://doi.org/10.1145/359340.359342), p. 120–126.

Rohloff K, C. D., 2014. A scalable implementation of fully Homomorphic encryption built on NTRU. *Financial Cryptography and Data Security,* 8438(https://doi.org/10.1007/978-3-662-44774-1_18), p. 221–234.

R, P., 2017. Parameter selection in lattice-based cryptography. *University of London. PhD Thesis, Royal Holloway.*

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. *Neural networks,* Volume 61, p. 85–117.

SD, G., 2002. Elliptic curve paillier schemes. *J. Cryptology,* 15(https://doi.org/10.1007/s00145-001-0015-6), p. 129–138.

T. Hastie, R. T. a. J. F., n.d. Unsupervised learning. *The elements of statistical learning,* Volume 2009, p. 485–585.

Tabian, I., Fu, H. & Sharif Khodaei, Z., 2019. A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. *Sensors,* 4933(https://doi.org/10.3390/s19224933).

Tchernykh A, S. U. T. E. B. M., 2019. Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. *J Comput Sci 36,* 100581(https://doi.org/10.1016/j.jocs.2016.11.011).

T, E., 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans Inf Theory,* 31(https://doi.org/10.1109/TIT.1985.1057074), p. 469–472.

T, R., 2020. Data protection in virtual environments. *(DPRIVE). DARPA/MTO, Technical report.*

Van Dijk M, G. C. H. S. V. V., 2010. Fully Homomorphic Encryption over the Integers. *Advances in Cryptology – EUROCRYPT,* 6110(https://doi.org/10.1007/978-3-642-13190-5_2), p. 24–43.

Vapnik, V. N., 1999. An overview of statistical learning theory. *IEEE transactions on neural networks,* Volume 10, p. 988–999.

V, V., 2011. Computing blindfolded: new developments in fully Homomorphic Encryption. *IEEE,* 52(https://doi.org/10.1109/FOCS.2011.98), p. 5–16.

Y. LeCun, L. B. Y. B. a. P. H., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* Volume 86, p. 2278–2324.

Y., Z., 1997. Digital signcryption or how to achieve cost(signature & encryption) ≪ cost(signature) + cost(encryption). *Kaliski B.S. (eds) Advances in Cryptology — CRYPTO,* 1294(https://link-springer-com.surrey.idm.oclc.org/chapter/10.1007%2FBFb0052234).

Yani, M. &. I. S. &. S. M., 2019. Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. *Journal of Physics: Conference Series.*

Z, B., 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. *Advances in Cryptology – CRYPTO 2012,* 7417(https://doi.org/10.1007/978-3-642-32009-5_50), p. 868–886.

Zheng Q, W. X. K. K. M. Z. W. G. B. G. W., 2018. A lightweight authenticated encryption scheme based on chaotic SCML for railway cloud service.. *IEEE Access,* Volume https://doi.org/10.1109/ACCESS.2017.2775038, p. 6: 711– 722.