SECURE MODEL VERIFICATION AND PRIVACY PRESERVATION WITH ZK-SNARKS AND NEURAL NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DURSUN OYLUM SERINER GERENLI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

SEPTEMBER 2023

Approval of the thesis:

SECURE MODEL VERIFICATION AND PRIVACY PRESERVATION WITH ZK-SNARKS AND NEURAL NETWORKS

submitted by **DURSUN OYLUM SERINER GERENLI** in partial fulfillment of the requirements for the degree of **Master of Science in Cryptography Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Selçuk Kestel
Dean, Graduate School of **Applied Mathematics** _______________

Assoc. Prof. Dr. Oğuz Yayla
Head of Department, **Cryptography** _______________

Prof. Dr. Ferruh Özbudak
Supervisor, **Mathematics, METU** _______________

Examining Committee Members:

Assist. Prof. Dr. Buket Özkaya
Cryptography, METU _______________

Prof. Dr. Ferruh Özbudak
Mathematics, METU _______________

Assist. Prof. Dr. Eda Tekin
Cryptography, Karabük University _______________

Date: _______________

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: DURSUN OYLUM SERINER GERENLI

Signature :

ABSTRACT

SECURE MODEL VERIFICATION AND PRIVACY PRESERVATION WITH ZK-SNARKS AND NEURAL NETWORKS

Seriner Gerenli, Dursun Oylum
M.S., Department of Cryptography
Supervisor : Prof. Dr. Ferruh Özbudak

September 2023, 79 pages

Neural networks are widely used learning models to achieve successful results in many application areas today. However, proving and sharing the accuracy and reliability of these networks is often limited due to privacy and security challenges. In this study, a method of cryptographic proving the accuracy of neural networks without revealing their intrinsic components is presented. The method is presented by using the Circom programming language to create a circuit containing these elements by making use of the final weights, bias values, and inputs of the neural networks. The use of the Circom programming language makes it possible to convert neural network elements into electronic circuits. The resulting circuit contains the representation of the neural network model and mimics the transformation from inputs to outputs. It is also used with Groth16 which is a Zero Knowledge Proof system to prove the accuracy of the neural network without leaking private information. As in this study, the newly produced circuit can be used with the help of zkREPL or terminal. As a result, an experimental method is presented to prove the real-world performance of the neural network model and increase the reliability of the model, and using the knowledge found in the literature, an approach has been explored to be implemented to solve current security problems. In this way, the correctness of the model can be proven without directly telling the hidden inputs to the other party.

Keywords: Cryptography, Zero Knowledge Proof, Neural Networks, Groth16, Circom, Machine Learning

ÖZ

ZK-SNARK VE SİNİR AĞLARI İLE GÜVENLİ MODEL DOĞRULAMA VE ÖZEL VERİNİN KORUNMASI

Seriner Gerenli, Dursun Oylum
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Prof. Dr. Ferruh Özbudak

Eylül 2023, 79 sayfa

Yapay sinir ağları günümüzde birçok uygulama alanında başarılı sonuçlara ulaşmak için yaygın olarak kullanılan öğrenme modelleridir. Ancak, bu ağların doğruluğunun ve güvenilirliğinin kanıtlanması ve paylaşılması genellikle gizlilik ve güvenlik sorunları nedeniyle sınırlıdır. Bu çalışmada, yapay sinir ağlarının içsel bileşenlerini ortaya çıkarmadan doğruluğunu kriptografik olarak kanıtlayan bir yöntem sunulmaktadır. Yöntem, sinir ağlarının nihai ağırlıkları, bias değerleri ve girdilerinden yararlanarak bu elemanları içeren bir devre oluşturmak için Circom programlama dili kullanılarak sunulmuştur. Circom programlama dilinin kullanılması, sinir ağı elemanlarının elektronik devrelere dönüştürülmesini mümkün kılar. Ortaya çıkan devre, sinir ağı modelinin temsilini içerir ve girdilerden çıktılara dönüşümü taklit eder. Ayrıca, sinir ağının doğruluğunu özel bilgileri sızdırmadan kanıtlamak için Zero Knowledge Proof sistemi olan Groth16 ile birlikte kullanılır. Bu çalışmada olduğu gibi yeni üretilen devre zkREPL veya terminal yardımıyla kullanılabilir. Sonuç olarak, sinir ağı modelinin gerçek dünya performansını kanıtlamak ve modelin güvenilirliğini artırmak için deneysel bir yöntem sunulmuş ve literatürde bulunan bilgiler kullanılarak mevcut güvenliğik probleminin çözümüne yönelik uygulanacak bir yaklaşım araştırılmıştır. Bu sayede gizli girdiler doğrudan karşı tarafa söylenmeden modelin doğruluğu kanıtlanabilmektedir.

Anahtar Kelimeler: Kriptografi, Sıfır Bilgi İspatı, Sinir Ağları, Groth16, Circom, Makine Öğrenmesi

To my family

ACKNOWLEDGMENTS

I would like to extend my heartfelt appreciation to my supervisor, Prof Dr. Ferruh Özbudak, for his help throughout the process of developing and preparing this thesis.

I would like to express my gratitude to the members of my examination committee, Assist. Prof. Dr. Buket Özkaya and Assist. Prof. Dr. Eda Tekin, for their valuable time dedicated to me.

I would like to thank my manager Ömer Yaminoğlu, who supported the writing of this thesis.

I would like to thank my dear friends who have always supported me, especially my high school friends who encouraged me during the writing of my thesis.

I would like to express my gratitude to my dear husband Umur Gerenli. He gave me great ideas, even on topics completely unrelated to his expertise. Thanks for being by my side with incredible suggestions and constant support.

I would want to extend my heartfelt gratitude to my family. They consistently provide me with their love and support whenever I need it.

TABLE OF CONTENTS

ABSTRACT vii

ÖZ ix

ACKNOWLEDGMENTS xiii

TABLE OF CONTENTS xv

LIST OF TABLES xix

LIST OF FIGURES xx

LIST OF ABBREVIATIONS xxi

CHAPTERS

1 INTRODUCTION 1

1.1 Related Works 2

1.2 The Outline 3

2 ZERO KNOWLEDGE PROOF 5

2.1 What is a zero-knowledge proof ? 5

2.1.1 Requirements for a zero-knowledge proof system specification 7

2.1.2 Specifying Statements for ZK 8

2.1.3 Example1: Discrete logarithm (discrete-log) 8

2.1.4 Example2: Graph non-isomorphism 9

2.1.5 Example3: Schnorr's Identification Protocol 10

2.1.6 Differences Between Interactive and Non-interactive ZKP 11

2.2 What is the ZK-snark? 12

2.2.1 Example of ZK-Snark 13

2.2.1.1 Preliminaries 13

2.2.1.2 Example Solution about ZK-SNARK . 16

2.2.2 Groth16 23

3 MACHINE LEARNING 33

3.1 Definition of Machine Learning 33

3.2 Supervised Learning and Unsupervised Learning 34

3.3 Process of Developing Model 34

3.4 Importance and relevance of privacy in ML 36

3.5 Zero Knowledge Machine Learning 39

3.6 Neural Networks 39

3.6.1 Neural Networks Structure with Python Code . . . 40

3.7 Reversing Model 51

4 IMPLEMENTATION OF ZKML PROTOTYPE 53

4.1 CIRCOM and Snarkjs 55

4.2 Terminal 55

4.2.1 Step 1: Installation 56

4.2.2 Step 2: Writing Circuits 57

4.2.3 Step 3: Computing Witness 61

4.2.4 Step 4: Proving circuits (Groth16) 62

4.3 ZkREPL 69

5 CONCLUSION 71

5.1 Limitations 72

5.2 Future Work 72

REFERENCES 75

LIST OF TABLES

Table 2.1 Example of ZKP 5

Table 2.2 Discrete Logarithm Protocol 8

Table 2.3 Graph Non-isomorphism Protocol 9

Table 2.4 Schnorr's Identification Protocol 10

LIST OF FIGURES

Figure 2.1 Arithmetic circuit for $f(s_1, s_2, s_3) = (s_1 \cdot s_2) \cdot s_3$ 13

Figure 2.2 Flattening for ZK-snarks 17

Figure 3.1 Process of Machine Learning [2] 36

Figure 3.2 Neuron [28] 40

Figure 3.3 Neural Netwrok Structure 41

Figure 3.4 Output of Python code 1 42

Figure 3.5 Output of Python code 2 43

Figure 3.6 Output of Python code 3 43

Figure 3.7 ReLU Graph 45

Figure 3.8 Output of Python code 5 46

Figure 3.9 Output of Python code 7 48

Figure 3.10 Train Loss and Validation Loss for each epoch 49

Figure 3.11 Train Accuracy and Validation Accuracy for each epoch 49

Figure 3.12 Confusion Matrix of Neutral Network Model 50

Figure 3.13 Output of Python code 8 51

Figure 4.1 Neutral Network Flowchart 54

Figure 4.2 Circom and Snarkjs 56

Figure 4.3 Image of Terminal Output 1 60

Figure 4.4 Image of Terminal Output 2 62

Figure 4.5 The output of zkREPL 69

LIST OF ABBREVIATIONS

CRS	Common Reference String
IZKP	Interactive Zero Knowledge Proof
ML	Machine Learning
NZKP	Non-Interactive Zero Knowledge Proof
QAP	Quadratic Arithmetic Programs
R1CS	Rank-1 Constraint System
ReLU	Rectified Linear Unit
ZK	Zero Knowledge
ZKML	Zero Knowledge Machine Learning
ZKP	Zero Knowledge Proof
ZKSNARK	Zero-Knowledge Succinct Non-interactive ARgument of Knowledge

CHAPTER 1

INTRODUCTION

The discipline of cryptography has traditionally been used to guarantee the privacy and integrity of information exchange. The use of these tools, initially limited to military and diplomatic communication, gradually extended to other domains of civilian life, eventually establishing itself as a fundamental component of the contemporary digital landscape. The proliferation of the Internet has heightened the significance of encryption in safeguarding personal data and sensitive information.

Machine learning is a pivotal technology that finds extensive use in several domains, including but not limited to data analysis, predictive modeling, and automated decision-making across diverse sectors. However, machine learning models are not exclusively dependent on the efficacy of algorithms and training data. The quality, integrity, and confidentiality of the input data play a critical role in protecting the performance and security of the model. Cryptography includes a collection of mathematical methodologies that facilitate the establishment of safe encryption and decryption mechanisms for data. Cryptography techniques provide a viable means of safeguarding input data or other value not to be shared. The reason for maintaining confidentiality about the model inputs is expanded further inside the thesis. Using proper data storage techniques enhances the dependability of the model and safeguards against the occurrence of inaccurate outcomes.

The main problems discussed in this thesis relate to the following. Firstly, the training data sample may include personal and private information. Individuals who own ownership of this dataset may be interested in using it to train a model while maintaining a sense of control and authority over the data. Secondly, a person with malicious motives possessing knowledge of the model's output may manipulate the model to get desired outcomes. Thirdly, it facilitates competitors in gaining awareness of the fundamental principles of powerful AI models and using this knowledge to their advantage. This thesis explores cryptographic techniques that may be used to minimize the occurrence of adverse events.

By using the groth16 algorithm, it will be proved that the predictions of the model belong to this model without revealing the weight, input and biases of the neural network model. For this, an application is presented via terminal and ZKML compiler as a method. It has been stated in the literature that ZK-snarks may be used with neural network models; however, there have been deficiencies in terms of their application, which is why Groth16 and Neural Network models was chosen for this thesis. This thesis will provide a model of its applicability because of these reasons.

1.1 Related Works

Zhang et al. (2020) conducted a research [38] with the objective of using the Zero Knowledge Proof technique to safeguard the privacy of a decision tree model. The primary objective of this research was to maintain the integrity of the forecasts and precision of the decision tree model.

In their academic article titled "VeriML: Enabling Integrity Assurances and Fair Payments for Machine Learning as a Service," [39] Zhao et al. (2021) try to develop a methodology using snark for protecting the training data of six distinct models. This study is conducted according with the goals of maintaining data integrity and facilitating equitable financial transactions.

"Decentralized Federated Machine Learning with Blockchain and Zero Knowledge Proofs" focuses on how Federated Machine Learning (FML) models can use Zero Knowledge Proofs (ZKP) and blockchain technologies to protect privacy without exposing the main data source [33].

The aforementioned research was carried out by Lee, Ko, Kim, and Oh (2020) and was subsequently published with the title "vCNN: Verifiable Convolutional Neural

Network based on zk-snarks". This article [14] presents a novel convolutional neural network (CNN) architecture, referred to as the accelerated verification CNN (vCNN), which aims to significantly enhance proof performance. The primary objective of this framework is to guarantee the verifiability of convolutional neural networks by the incorporation of zk-SNARKs technology. This paper introduces a novel and fast language for enhancing the proving performance of convolution equations.

In the article [7] is a publication titled "SafetyNets: Verifiable Execution of Deep Neural Networks on an Untrusted Cloud," Ghodsi, Gu, and Garg introduce a conceptual framework that aims to establish the reliability of deep learning model inference tasks conducted by an untrusted cloud service provider on behalf of its client in 2017.

1.2 The Outline

This thesis starts with the basic principles of Zero-Knowledge Proof (ZKP) and discusses the applications of this concept in different fields [5], [22]. To understand the essence of ZKP, examples of number theory such as discrete logarithm, graph non-isomorphism, and Schnorr's Authentication Protocol are examined [31]. Also, both interactive and non-interactive ZKP approaches are introduced [9].

Building on this basic insight, the focus is on ZKsnarks and specifically the Groth16 algorithm [6], [20], [36]. By giving examples about the Groth16 algorithm [27], examples of data that protects confidentiality, which is one of the powerful and wide application areas of ZKP, are shown.

The focus of the thesis then shifts to Machine Learning(ML) [21]. Firstly, the concepts of supervised and unsupervised learning are introduced [2], [26], [29]. Then, the development processes of a model and the importance of confidentiality in this process are emphasized [30]. It also explains Zero Knowledge Machine Algorithm (ZKML) [13], [25].

After the definition of neural network models [11], [32], [37] it is explained how a model is developed over a Python example [5], [16], [18]. This model demonstrates how weights, biases, and inputs are validated without disclosure, using the Groth16 algorithm and using a terminal or online compiler (ZkREPL) [3], [12], [29]. This highlights how ZKP can be integrated with machine learning and how data privacy can be protected.

CHAPTER 2

ZERO KNOWLEDGE PROOF

2.1 What is a zero-knowledge proof ?

According to "ZKProof Community Reference" study [5] (Benarroch et al., 2022), ZKP requires proving true statements without saying secret information directly. The prover tries to prove the statement, and the verifier continuously checks the results. In addition, a statement (Notation: $x \in L$) means that it is a claim, and the prover and verifier know it. The statement needs to be substrate and call instance (Notation: I). Then, the secret information calls the witness (Notation: w). This part focuses on some examples of ZKP in real life, and the above table shows some examples,

Table 2.1: Example of ZKP

Statement	Instance	Witness
I am an adult	Tamper-resistant identification chip	Birthdate and personal data
We are not bankrupt	Encrypted bank records	Portfolio data-decryption key
This expression is a theorem	The logical rules of inference	Logical implications

Another example is the famous Ali Baba Cave. The exploration of Ali Baba Cave provides valuable insights into the comprehension of ZKP. For clarity, the story subjects could be Peggy and Victor. This is the story of Peggy and Victor, who found a ring-shaped cave with a closing magic door at the entry on the other side. She says she's found a secret word that opens the magical door but doesn't want to tell him what it is. He also wants to find out if Peggy does know the secret word. After that, Peggy and Victor agree to call the two ways "Paths A" and "Paths B." Then she goes in any direction without him being able to see where she is going. After that, he goes into the cave and shouts the name of either the A or B way he wants her to take to get back. Knowing the secret word, she can find the right path back to him without trouble. Due to this, Peggy and Victor can do this action more than once. He is more sure that she knows the secret word as he says it repeatedly.

- **Complete:** The prover tries to show the correct proof of the statement, which the verifier will accept. This property means that if the statement being proved is confirmed, an honest prover can convince an honest verifier of its truth with a high probability. The evidence system will be complete if a prover and a valid witness can convince an honest prover that this statement is true. Now, the prof system is considered. In the exercise that follows, imagine an adversary who is complete.

  1. Run Setup(params) $\longrightarrow$ (setupR, setupP , setupV , auxi)
  2. The adversary select a worst case instance and witness Adversary(params, setupR, setupP , setupV , auxi)$\longrightarrow$ (x,w)
  3. The interaction continues until Prover gives a error or verifier gives reject or accept the confirmation. If the protocol does not end, it will error. $\langle$ Prove(setupP , x,w, start) ; Verify(setupV , x)$\rangle$ $\longrightarrow$ result
  4. Adversary wins if (setupR, x,w) $\in$ R and result is not accept.

  If no one ever creates an effective adversary with a significant advantage, a proof system for R operating on params is finished. The application will determine what constitutes an effective adversary (computer hardware, operating time, memory utilization, lifetime, incentives, etc.) and how much benefit can be allowed. Statistical completeness (also known as unconditional completeness), where the chance of success is low for any opponent, and perfect completeness, where the advantage is precisely zero for any adversary, are examples of exceptionally strong cases.

- **Soundness:** The prover tries to show proof of the statement, but it is not true, and the verifier will reject it. This property means that if the statement being proved is false, no cheating prover can convince an honest verifier that it is true with a high probability. A proof system is sound if a dishonest verifier has little to no chance of being persuaded that a misleading statement is true by a dishonest prover. A proof system's complete specification must include an exact definition of soundness that conveys this intuition.

1. Run Setup(params) $\longrightarrow$ (setupR, setupP , setupV , auxi)
2. The adversary select instance Adversary(params, setupR, setupP , setupV , auxi)$\longrightarrow x$
3. The adversary interact with the verifier $\langle$ Adversary ; Verify(setupV , x)$\rangle$ $\longrightarrow$ result
4. Adversary wins if (setupR, x) $\notin L$ and result is accept.

- **Zero-knowledge:** The secret information will always be secret, meaning the verifier cannot know it. This property means that the protocol does not reveal any information about the secret beyond that the statement being proved is true. In other words, the verifier learns nothing about the secret itself.

$$\text{Advantage(params) = Pr[Adversary wins] - } \frac{1}{2}$$

2.1.1 Requirements for a zero-knowledge proof system specification

The proof system has some requirements,

1. The type of statement definition must be clear.
2. Algorithm details are known by prover and verifier and all construction about this system.
3. Setup definitions used by the prover and verifier. ("$PrivateSetup_P$ " or "$PrivateSetup_V$", respectively not known to the other party)
4. The primary objective of the proof system is to provide precise and well-defined specifications of the security it aims to provide.
5. The security analysis of the ZKP system and list of assumptions that have not yet been confirmed.

2.1.2 Specifying Statements for ZK

The statements considered in this study fit into the relation R between instances x and witnesses w. The relation R specifies which pairs (x, w). Then, they are considered related to each other, but they are not. L defines that instance x has a witness w in R. These notes mention details about definitions.

The statement of form is $x \in L$, this means that "I know witness w of instance x in R". Acceptance means that $(x, w) \in R$, while rejection means that $(x, w) \in L$ and that (w) is not a witness to $(x, w) \in R$.

2.1.3 Example1: Discrete logarithm (discrete-log)

The following is an explanation by step of the (ZKP) protocol for showing knowledge of a discrete logarithm.

- let p be large number and $p = 2q + 1$ and q is prime.
- g be generator of the group $\mathbb{Z}_p{}^* = \{1, ..., p-1\} = \{g^i : i = 1, ..., p-1\}$
- w be secret information known by prover
- $x = g^w(\mod p$ be the instance known both of them
- P: I know base g of the x, mod p
- Relation is $R = \{(x, w) : g^w(\mod p)\}$
- Language is $L = \{x : \exists w : (x, w) \in R$

This table explains the protocol briefly, and g, x, p publish.

Table 2.2: Discrete Logarithm Protocol

Verifier	Instance	Prover
		Choose a random number x
		Compute $h = g^r$
	h $\longleftarrow$	
Choose a random number b		
	$\longrightarrow$ b	
		Compute $s = (r + bw) \mod p - 1$
	s $\longleftarrow$	
$x^s \mod p = hg^b \mod p$??		

Focus on the how check use this method.

$$x^{(r+bw)} = x^r x^{bw} = x^r x^{wb} = hg^b$$

2.1.4 Example2: Graph non-isomorphism

The common inputs G_1 and G_2 are not isomorphism.

Table 2.3: Graph Non-isomorphism Protocol

Verifier	Instance	Prover
		Choose a random number $i \in 1, 2$
		Randomly select permutation $\pi \in S_{V_i}$
		Compute $F = \pi(G_i)$
	F $\longleftarrow$	
Find $j \in (1, 2)$		
G_1 , F or G_2 , F is isomorphism		
	$\longrightarrow$ j	
		check j=i ?

π is the secret value. Also, the set is not isomorphic groups and the secret value belongs it.

2.1.5 Example3: Schnorr's Identification Protocol

In "Cryptography Made Simple" book (Smart, 2016) [31] shows this algorithm with its details. Peggy knows the x such that $y = g^x$. During creating an identification protocol, i.e the wish is Peggy to show in ZK that she knows the value of x.

Table 2.4: Schnorr's Identification Protocol

Peggy		Victor
knows x		knows y
$r \leftarrow g^k$ for a random $k \leftarrow \mathbb{Z}/q\mathbb{Z}$	$\longrightarrow r$	
	$e \longleftarrow$	$e \leftarrow \mathbb{Z}/q\mathbb{Z}$
$s \leftarrow k + (x * e)\ (mod\, q)$	$\longrightarrow s$	$r = g^s * y^{-e}$

- Peggy generates a random k from $\mathbb{Z}/q\mathbb{Z}$ and she computes the r value such that $r = g^k$.
- Peggy sends the r to Victor.
- Victor generates the random e from $\mathbb{Z}/q\mathbb{Z}$.
- Victor sends the e to Peggy.
- Peggy computes the s such that $s = k + (x * e)$.
- Peggy sends the s to Victor.
- Victor verifies $r = g^s * y^{-e}$.

Now, the ZKP features Completeness, Soudness and Zero Knowledge are displayed on this protocol.

- **Completeness**: If Peggy actually knows the secret discrete logarithm x, Victor can accept the protocol since,

$$g^s * y^{-e} = g^{k+x*e} * (g^x)^{-e} = g^{k+x*e} * g^{-x*e} = g^k = r$$

- **Soundness**: Let us assume that Peggy does not know the secret discrete logarithm x, she has to choose a random value from $\mathbb{Z}_q$, meaning that the probability of Peggy convincing Victor is $\frac{1}{q}$.

- **Zero-Knowledge:** "What did Victor learn from the protocol?" is the question. If Victor can write proper protocol without talking to Peggy, he can't make anyone believe that Peggy knows something. Here, the same thing happens. Victor can write the following as an acceptable protocol:

- $e \leftarrow \mathbb{Z}/q\mathbb{Z}$.
- $r = g^s * y^{-e}$.
- Output the transcript,

$$
\begin{aligned}
&P \rightarrow V : r, \\
&V \rightarrow P : e, \\
&P \rightarrow V : s.
\end{aligned}
$$

One must notice how easy it is to write a valid protocol without any interaction, meaning that no one can understand if the transcript given above is a simulation or not. Therefore, Schnorr's Identification Protocol has the ZK property.

2.1.6 Differences Between Interactive and Non-interactive ZKP

There are two distinct categories of ZKP protocols, namely interactive and non-interactive protocols. To prove a statement using an interactive ZKP (IZKP) protocol, both parties (prover and verifier) must be online. A non-interactive ZKP (NZKP) protocol, on the other hand, enables the prover to prove the assertion whether the verifier is online or offline. In other words, with NZKP protocols, the verifier may verify the claim without interacting with the prover online. As a result, a NZKP protocol is often quicker and more efficient since it does not need any online connection or interaction between the prover and the verifier to prove the statement or claim. A NZKP protocol, on the other hand, demands that both the verifier and the prover share a random string, usually given by a trusted third party. A pre-planned application of this random string is also necessary.

2.2 What is the ZK-snark?

The notion of SNARK, or concise non-interactive argument of knowledge, is especially important in the domain of non-interactive proofs for showing the integrity of outcomes for huge computations. These words refer to a proof system, which is:

- **Succient:** The proof is quite little in comparison to the size of the statement or witness, i.e. the size of the calculation itself.
- **Non-interactive:** It is not necessary and interaction rounds between the prover and the verifier.
- **Argument:** It is secure only for provers with constrained computing resources, which implies that provers with adequate computational power may persuade the verifier of a false assertion.
- **Knowledge-sound:** The prover is only able to generate proof if possessing knowledge of a particular witness for the statement. In an official capacity, for every prover capable of creating a valid proof, there exists an extractor capable of extracting a witness, sometimes referred to as "the knowledge," for the statement.

SNARK systems may also have a zero-knowledge feature, which allows the proof to be performed without exposing anything about the intermediate stages (the witness).

A zk-SNARK protocol, like any other non-interactive proof system, is defined by three algorithms that operate in the following approach:

- The method Gen serves as the setup process, producing a string that contains essential setting information for subsequent steps in the proving process. Additionally, it generates a verification key, which is often considered to be known only to the verifier. Typically, the operation is overseen by a reliable entity.
- The algorithm "Prove" is a proving algorithm that accepts as input the setting information, the assertion, and a matching witness, and produces the proof as output.

- The Verify algorithm is designed to accept or reject a proof based on the verification key, statement, and proof provided as input. It returns a value of 1 to indicate acceptance or 0 to indicate rejection.

2.2.1 Example of ZK-Snark

This example is that of Vitalik Buterin, the founder of Ethereum. Vitalik Buterin's "Quadratic Arithmetic Programs: from Zero to Hero" paper [34] (Vitalik, 2016) gives a comprehensive and elaborated solution below. However, it is crucial to begin by providing the essential introductory details. In order to properly understand this specific example, it is necessary to have some foundational information.

2.2.1.1 Preliminaries

Definition of Arithmetic Circuit: This is an acyclic graph and it includes nodes which are addition and multiplication gates. Then, wires connect inputs and outputs by using nodes. Each node has two inputs and one output. The figure below explains it.

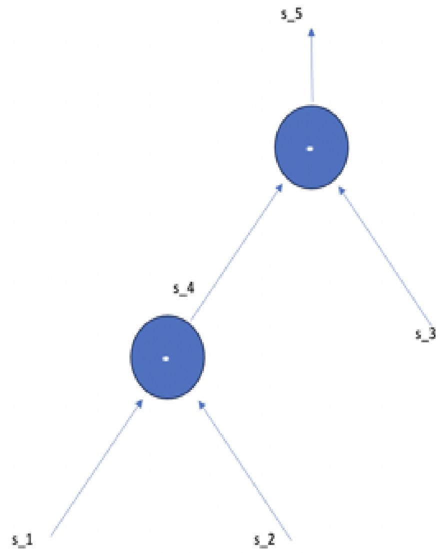


Figure 2.1: Arithmetic circuit for $f(s_1, s_2, s_3) = (s_1 \cdot s_2) \cdot s_3$

The witness $s = (s_1, s_2, ..., s_n)$ to be the values for the n wires so that the inputs and outputs of each gate match the requirements set by the gate.

Rank-1 Constraint System (R1CS): The 1 refers to the matrix of degrees. It uses the representation of the structure. There are three triplets (a, b, c); they are vectors. There is an equation and its form,

$$(A) \cdot (B) - (C) = 0$$

$(A), (B), (C)$ is a linear combinations then,

$$(A) = a_1 \cdot s_1 + a_2 \cdot s_2 + ...$$

$$(B) = b_1 \cdot s_1 + b_2 \cdot s_2 + ...$$

$$(C) = c_1 \cdot s_1 + c_2 \cdot s_2 + ...$$

s_i denotes solution s vector.
For R1CS, outputs known by both of parties, solution vector is checked by verifier and prover.

Quadratic Arithmetic Programs (QAP): There are target point $r_1, r_2, ...r_m \in \mathbb{F}_p$. Then, compute $t(x) = \prod_{Q=1}^{m}(x - r_q)$

$$u_i(r_q) = u_{i,q}$$

$$v_i(r_q) = v_{i,q}$$

$$w_i(r_q) = w_{i,q}$$

In the circuit, each goal point is like a gate. The QAP builds $3n$ polynomials for each target point r_q that, when evaluated at r_q, give the $3n$ constants of the qth gate's rank-1-constraint. The m rank-1 limits are written as a single equation over the functions in the QAP.

Lagrange Polynomials: There are some ways to find the unique polynomial $p_n(x)$ of degree n that makes the statement $p_n(x_i) = y_i$, where $i = 0, 1, ..., n$ given the points (x_i, y_i). Interpolation points are the points $x_0, x_1, ..., x_n$.

The values $(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$ are said to be interpolated by the polynomial $p_n(x)$.

If the interpolation points $x_0, ..., x_n$ are all different, then finding a polynomial that goes through the points (x_i, y_i), where $i = 0, ..., n$ is the same as computing linear equations $Ax = b$ that has only one solution.

In Lagrange interpolation, the matrix A is just the identity matrix because the interpolating equation is:

$$p_n(x) = \sum_{j=0}^{n} y_i \mathcal{L}_{n,j}(x)$$

Consider the $\mathcal{L}_{n,j}(x)$:

$$\mathcal{L}_{n,j}(x) = \begin{cases} 1 & if \quad i = j \\ 0 & if \quad i \neq j \end{cases}$$

Finally,

$$\mathcal{L}_{n,j}(x) = \prod_{k=0, k\neq j}^{n} \frac{x - x_k}{x_j - x_k}$$

The next finding shows that Lagrange polynomials can be used to solve the problem of polynomial interpolation.

Eliptic Curve: The equation for an elliptic curve in two variables is a particular type of cubic equation. A curve with this form is called an "elliptic curve",

$$y^2 = x^3 + Ax + B$$

Addition. The second intersection of the tangent of P at another point is used. The symmetry is taken from the x-axis of the point where the tangent intersects.

$$P(x_1, y_1), Q(x_2, y_2)$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

$$P \oplus Q = R(x_3, y_3)$$

$$x_3 = (\lambda^2 - x_1 - x_2)$$

$$y_3 = (\lambda(x_1 - x_3) - y_1)$$

Doubling. In cases when the points P and Q coincide, the process of addition is altered due to the absence of a clearly defined straight line passing through point P. Consequently, the operation is completed by using a limiting case, namely the tangent to the curve at point P, denoted as E.

$$\lambda = \frac{3{x_1}^1 + 2}{2y_1}$$

$$P \oplus P = R(x_3, y_3)$$

$$x_3 = (\lambda^2 - 2x_1)$$

$$y_3 = (\lambda(x_1 - x_3) - y_1)$$

2.2.1.2 Example Solution about ZK-SNARK

In this example, the secret value equals 3 and solution steps are given below step by step.

Problem Statement $\longrightarrow$ R1CS

This is the statement:

$$x^3 + x + 5 = 35$$

First, the flattening process is applied. In this porcess, convert the original code which may have very complicated statements and expression and two types statements.

- $x = y$ (where y can be a variable or a number)
- $x = y(op)z$ (where op can be +, -, *, or / and y and z can be variables, numbers or expressions)

The result of the flattening:

$$a = x * x \to x^2$$

$$b = a * x \to x^3$$

$$c = b + x \to x^3 + x$$

$$d = c + 5 \to x^3 + x + 5$$

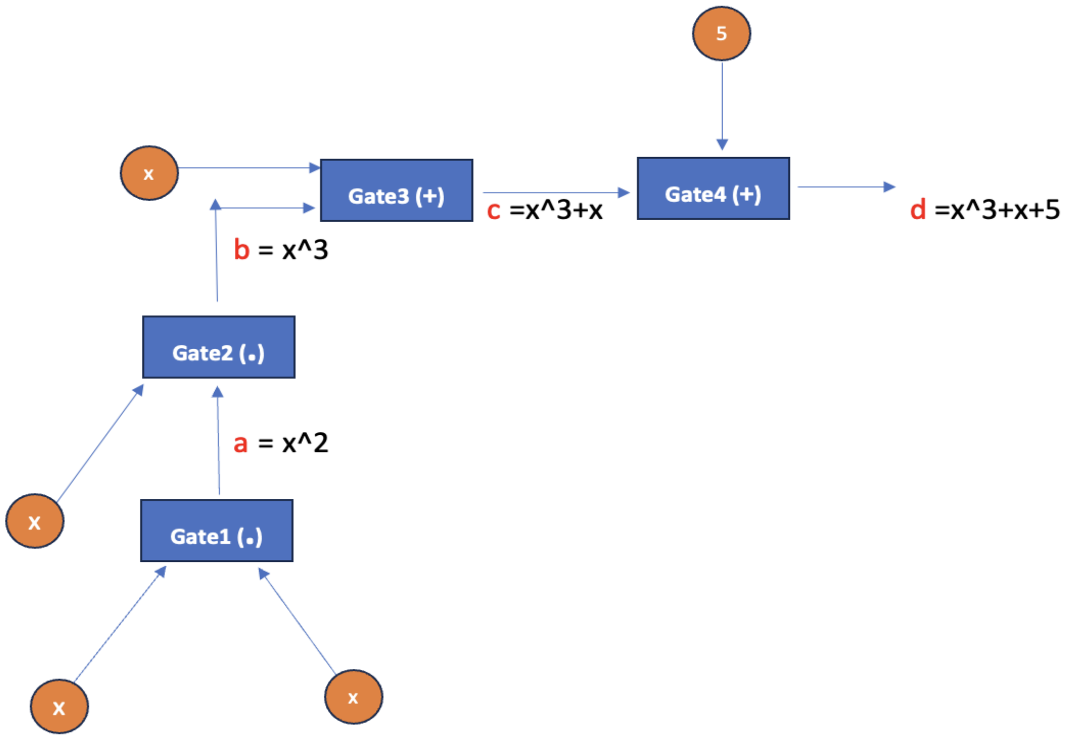


Figure 2.2: Flattening for ZK-snarks

This is the solution:

$$s = [1, x, a, b, c, d], \quad secret : [1, 3, 9, 27, 30, 35]$$

An R1CS has three-vector groups (A, B, C). The s must be satisfy $s{\cdot}a{*}s{\cdot}b{-}s{\cdot}c = 0$.

Each gate has triple (A, B, C):

Gate1: $a = x * x = x^2$

$s = [1, x, a, b, c, d]$

$A = [0, 1, 0, 0, 0, 0]$

$B = [0, 1, 0, 0, 0, 0]$

$C = [0, 0, 1, 0, 0, 0]$

Gate2: $b = a * x$

$s = [1, x, a, b, c, d]$

$A = [0, 0, 1, 0, 0, 0]$

$B = [0, 1, 0, 0, 0, 0]$

$C = [0, 0, 0, 1, 0, 0]$

Gate3: $c = b + x$

$s = [1, x, a, b, c, d]$

$A = [0, 1, 0, 1, 0, 0]$

$B = [1, 1, 0, 0, 0, 0]$

$C = [0, 0, 0, 0, 1, 0]$

Gate4: $d = c + 5$

$s = [1, x, a, b, c, d]$

$A = [5, 0, 0, 0, 0, 1]$

$B = [1, 0, 0, 0, 0, 0]$

$C = [0, 0, 0, 0, 0, 1]$

R1CS$\longrightarrow$ QAP

The next step is to convert this R1CS into QAP form, which uses the same reasoning but uses polynomials instead of dot products. The four groups of three vectors with lengths of six to six groups of three polynomials with degree 3. One of the requirements is that the polynomials must be evaluated at each x point. If the polynomials at $x = 1$ are computed, the first set of vectors is obtained, and if the polynomials at $x = 2$ are computed, the second set of vectors is received, and so on.

Gate1

$$\begin{array}{lll}
A_1(1) = 0 & B_1(1) = 0 & C_1(1) = 0 \\
A_2(1) = 1 & B_2(1) = 1 & C_2(1) = 0 \\
A_3(1) = 0 & B_3(1) = 0 & C_3(1) = 1 \\
A_4(1) = 0 & B_4(1) = 0 & C_4(1) = 0 \\
A_5(1) = 0 & B_5(1) = 0 & C_5(1) = 0 \\
A_6(1) = 0 & B_6(1) = 0 & C_6(1) = 0
\end{array}$$

Gate2

$$\begin{array}{lll}
A_1(2) = 0 & B_1(2) = 0 & C_1(2) = 0 \\
A_2(2) = 0 & B_2(2) = 1 & C_2(2) = 0 \\
A_3(2) = 1 & B_3(2) = 0 & C_3(2) = 0 \\
A_4(2) = 0 & B_4(2) = 0 & C_4(2) = 1 \\
A_5(2) = 0 & B_5(2) = 0 & C_5(2) = 0 \\
A_6(2) = 0 & B_6(2) = 0 & C_6(2) = 0
\end{array}$$

Gate3

$$\begin{array}{lll}
A_1(3) = 0 & B_1(3) = 1 & C_1(3) = 0 \\
A_2(3) = 1 & B_2(3) = 0 & C_2(3) = 0 \\
A_3(3) = 0 & B_3(3) = 0 & C_3(3) = 0 \\
A_4(3) = 1 & B_4(3) = 0 & C_4(3) = 0 \\
A_5(3) = 0 & B_5(3) = 0 & C_5(3) = 1 \\
A_6(3) = 0 & B_6(3) = 0 & C_6(3) = 0
\end{array}$$

Gate4

$$\begin{array}{lll}
A_1(4) = 5 & B_1(4) = 1 & C_1(4) = 0 \\
A_2(4) = 0 & B_2(4) = 0 & C_2(4) = 0 \\
A_3(4) = 0 & B_3(4) = 0 & C_3(4) = 0 \\
A_4(4) = 0 & B_4(4) = 0 & C_4(4) = 0 \\
A_5(4) = 1 & B_5(4) = 0 & C_5(4) = 0 \\
A_6(4) = 0 & B_6(4) = 0 & C_6(4) = 1
\end{array}$$

Lagrange interpolation will be used to transform the R1CS. Using Lagrange interpolation to take the first value from each vector, make a polynomial from that, process the first value of each vector, and then convert this to the second values, third values, etc., is repeated.

A polynomials

$A_1(1) = 0 \quad A_1(2) = 0 \quad A_1(3) = 0 \quad A_1(4) = 0$

$A_1(x) = -5 + 9.166x - 5x^2 + 0.833x^3$

$A_2(1) = 1 \quad A_2(2) = 0 \quad A_2(3) = 1 \quad A_2(4) = 0$

$A_2(x) = 8 - 11.333x + 5x^2 - 0.666x^3$

$A_3(1) = 0 \quad A_3(2) = 1 \quad A_3(3) = 0 \quad A_3(4) = 0$

$A_3(x) = -6 + 9.5x - 4x^2 + 0.5x^3$

$A_4(1) = 0 \quad A_4(2) = 0 \quad A_4(3) = 1 \quad A_4(4) = 0$

$A_4(x) = 4 - 7x + 3.5x^2 - 0.5x^3$

$A_5(1) = 0 \quad A_5(2) = 0 \quad A_5(3) = 0 \quad A_5(4) = 1$

$A_5(x) = -1 + 1.833x - x^2 + 0.166x^3$

$A_6(1) = 0 \quad A_6(2) = 0 \quad A_6(3) = 0 \quad A_6(4) = 0$

$A_6(x) = 0$

B polynomials

$B_1(1) = 0 \quad B_1(2) = 0 \quad B_1(3) = 1 \quad B_1(4) = 1$

$B_1(x) = 3 - 5.166x + 2.5x^2 - 0.333x^3$

$B_2(1) = 1 \quad B_2(2) = 1 \quad B_2(3) = 0 \quad B_2(4) = 0$

$B_2(x) = -2 + 5.166x - 2.5x^2 + 0.333x^3$

$B_3(1) = 0 \quad B_3(2) = 0 \quad B_3(3) = 0 \quad B_3(4) = 0$

$B_3(x) = 0$

$B_4(1) = 0 \quad B_4(2) = 0 \quad B_4(3) = 0 \quad B_4(4) = 0$

$B_4(x) = 0$

$B_5(1) = 0 \quad B_5(2) = 0 \quad B_5(3) = 0 \quad B_5(4) = 0$

$B_5(x) = 0$

$B_6(1) = 0 \quad B_6(2) = 0 \quad B_6(3) = 0 \quad B_6(4) = 0$

$B_6(x) = 0$

C polynomials

$C_1(1) = 0 \quad C_1(2) = 0 \quad C_1(3) = 0 \quad C_1(4) = 0$

$C_1(x) = 0$

$C_2(1) = 0 \quad C_2(2) = 0 \quad C_2(3) = 0 \quad C_2(4) = 0$

$C_2(x) = 0$

$C_3(1) = 1 \quad C_3(2) = 0 \quad C_3(3) = 0 \quad C_3(4) = 0$

$C_3(x) = 4 - 4.333x + 1.5x^2 - 0.166x^3$

$C_4(1) = 0 \quad C_4(2) = 1 \quad C_4(3) = 0 \quad C_4(4) = 0$

$C_4(x) = -6 + 9.5x - 4x^2 + 0.5x^3$

$C_5(1) = 0 \quad C_5(2) = 0 \quad C_5(3) = 1 \quad C_5(4) = 0$

$B_5(x) = 4 - 7x + 3.5x^2 - 0.5x^3$

$C_6(1) = 0 \quad C_6(2) = 0 \quad C_6(3) = 0 \quad C_6(4) = 1$

$\mathrm{C}_6(x) = -1 + 1.833x - x^2 + 0.166x^3$

Find $A \cdot s, B \cdot s$ and $C \cdot s$

$A \cdot s$

$$\begin{bmatrix} -5 & 8 & -6 & 4 & -1 & 0 \\ 9.166 & -11.333 & 9.5 & -7 & 1.833 & 0 \\ -5 & 5 & -4 & 3.5 & -1 & 0 \\ 0.833 & -0.666 & 0.5 & -0.5 & 0.166 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ 9 \\ 27 \\ 30 \\ 35 \end{bmatrix} = \begin{bmatrix} 43 \\ -73.343 \\ 38.5 \\ -5.166 \end{bmatrix}$$

$B \cdot s$

$$\begin{bmatrix} 3 & -2 & 0 & 0 & 0 & 0 \\ -5.166 & 5.166 & 0 & 0 & 0 & 0 \\ 2.5 & -2.5 & 0 & 0 & 0 & 0 \\ -0.333 & 0.333 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ 9 \\ 27 \\ 30 \\ 35 \end{bmatrix} = \begin{bmatrix} -3 \\ 10.332 \\ -5 \\ 0.666 \end{bmatrix}$$

$C \cdot s$

$$\begin{bmatrix} 0 & 0 & 4 & -6 & 4 & -1 \\ 0 & 0 & -4.333 & 9.5 & -7 & 1.833 \\ 0 & 0 & 1.5 & -4 & 3.5 & -1 \\ 0 & 0 & -0.166 & 0.5 & -0.5 & 0.166 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ 9 \\ 27 \\ 30 \\ 35 \end{bmatrix} = \begin{bmatrix} -41 \\ 71.666 \\ -24.5 \\ 2.833 \end{bmatrix}$$

Now, $A \cdot s * B \cdot s - C \cdot s = t$:

$$t = [-88.0, 592.666, -1063.777, 805.833, -294.777, 51.5, -3.444]$$

In order to address each restriction, a specific point is selected. This selection leads to the generation of four constraints, denoted as examples 1, 2, 3, and 4. These examples are used to construct a polynomial that can effectively execute remainderless division. minimal polynomial Z:

$$Z = (x - 1) * (x - 2) * (x - 3) * (x - 4)$$

$$Z = [24, -50, 35, -10, 1]$$

And find the h:

$$h = t/Z = [-3.666, 17.055, -3.444]$$

Finally, there is no reminder and reach the final result. Prover can send the h, t and Z, then Verifier can be verify by checking $A \cdot s * B \cdot s - C \cdot s = h * Z$.

2.2.2 Groth16

Daniel Jens Groth's 2016 work "On the Size of Pairing-based Non-interactive Arguments" [10] describes the Groth16 cryptography proof method. It is a zero-knowledge proof system and one of the most common zkSNARK proving methods.

Choose an elliptic curve pairing such that G_1, G_2, G_3 and generators g_1, g_2, g_3. And, these groups of order is 13. Then, pairing $e : G_1 \times G_2 \longrightarrow G_3$. This is the Groth16 protocol,

- Setup Phase: (CRS,ST) ← SETUP(R): R is a R1CS input and computes a Common Reference String (CRS) and Simulation Trapdoor (ST).
- Prover Phase: $\pi \leftarrow$ PROVE(R,CRS,I,W): I is instance and w is witness vector, algorithm prove takes them and output is proof.
- Verify: $accept, reject \leftarrow$ VFY(R,CRS,I,π): The algorithm result is reject or accept.

Circuit

Before you can check your program in Groth16, you have to change it into an R1CS constraint system.

1. R1CS

The witness vector for a rank-1 constraint system (R1CS) with n variables, m constraints, and p public inputs is $w \in \mathbb{F}_n$. By default, the first p values of w are the public input, and the first public input, w_0, is always 1. The m constraints in R1CS are a product equation between three inner products:

$$(w \cdot a_i) \cdot (w \cdot b_i) = w \cdot c_i$$

where vectors $(a_i, b_i, c_i) \in \mathbb{F}^{3 \cdot n}$.

2. Polynomials (QAP)

The choosen base $x \in \mathbb{F}^m$ and define functions for each condition so that $A_i(x_j) = A_{ij}$, and the same goes for B and C. Then define $A(X) = \sum_{i\in[0,n)} w_i \cdot A_i(X)$, like B and C. The QAP can be shown to be true if there is a low-degree $H(X)$ that satisfies,

$$A(X) \cdot B(X) - C(X) = H(X) \cdot Z_x(X)$$

It can only be considered "exact" (more accurately, a low-degree polynomial) if the conditions match.

In this section, a very useful example from the following document [27] will be shown.

Trusted setup

Groth16 needs to be set up in a certain way to make the common reference string, which is a set of numbers that everyone can use. This setup can be broken down into two parts: one that is *generic* and one that is *specific* to the circuit.

Also, the common reference string (CRS) model is based on the idea that there is a reliable setup in which everyone has access to the same string CRS from some distribution D. Schemes that have been shown to be safe in the CRS model are safe as long as the setup was done right. The common reference string model has proven to be a very easy way to build a wide range of efficient primitives that need to be very secure. This doesn't answer the question of how to set up what needs to be done. In reality, this is done by putting together a trusted setup process with multiple players using multi-party computation between users who are not thought to be working together.

Setup Stage:

Choose a maximum number of limits, m. Randomly generate the numbers $\alpha, \beta, \gamma, \delta$. In addition, example will be made with real numbers so that it will be more understandable. Example elliptic curve is,

$$BLS6_6 := \{(x, y)|y^2 = x^3 + 6 \quad for \quad all \quad x, y \in \mathbb{F}_{43}\}$$

This is finite cyclic group and there is 13 points,

$$\begin{aligned}G_1[13] =\{&(13,15) \rightarrow (33,34) \rightarrow (38,15) \rightarrow (35,28)\\ &\rightarrow (26,34) \rightarrow (27,34) \rightarrow (27,9) \rightarrow (26,9) \rightarrow\\ &(35,15) \rightarrow (38,28) \rightarrow (33,9) \rightarrow (13,28) \rightarrow Q\}\end{aligned}$$

$$\begin{aligned}G_2[13] =\{&(7v^2,16v^3) \rightarrow (10v^2,28v^3) \rightarrow (42v^2,16v^3) \rightarrow (37v^2,27v^3)\\ &\rightarrow (16v^2,28v^3) \rightarrow (17v^2,38v^3) \rightarrow (17v^2,15v^3)\\ &\rightarrow (16v^2,15v^3) \rightarrow (37v^2,16v^3) \rightarrow (42v^2,27v^3)\\ &\rightarrow (42v^2,27v^3) \rightarrow (10v^2,15v^3) \rightarrow (7v^2,27v^3) \rightarrow Q\}\end{aligned}$$

Find the target polynomial is $Z(x)$. Then, $m_1 = 5$ for first constraint and $m_2 = 7$ for second constraint in $\mathbb{F}_{13}$.

$$Z(x) = (x - m_1)(x - m_2)$$

$$Z(x) = (x - 5)(x - 7)$$

$$Z(x) = x^2 + x + 9$$

QAP result:

$A_0(x) = 0$	$B_0(x) = 0$	$C_0(x) = 0$
$A_1(x) = 0$	$B_1(x) = 0$	$C_1(x) = 7x + 4$
$A_2(x) = 6x + 10$	$B_2(x) = 0$	$C_2(x) = 0$
$A_3(x) = 0$	$B_3(x) = 6x + 10$	$C_3(x) = 0$
$A_4(x) = 0$	$B_4(x) = 7x + 4$	$C_4(x) = 0$
$A_5(x) = 7x + 4$	$B_5(x) = 0$	$C_5(x) = 6x + 10$

In the setup phase, a sampling process is conducted to get five random and invertible components, denoted as α, β, γ, δ, and τ, from the scalar field $\mathbb{F}_r$. Moreover, the simulation trapdoor is often known as the toxic waste of the initial configuration stage.

$$ST = (\alpha, \beta, \gamma, \delta, \tau)$$

Then, the τ is secret point for example $ST = (6, 5, 4, 3, 2)$ and $n = 1$, $m = 4$

$$CRS_{G_1} = \begin{cases} {g_1}^{\alpha}, {g_1}^{\beta}, {g_1}^{\delta}, \left({g_1}^{s^j}, ...\right)_{j=0}^{deg(Z)-1}, \left(g_1^{\frac{\beta\cdot A_j(s)+\alpha\cdot B_j(s)+C_j(s)}{\gamma}}\right)_{j=0}^{n}, \\ \left(g_1^{\frac{\beta\cdot A_{j+1}(s)+\alpha\cdot B_{j+1}(s)+C_j(s)}{\delta}}\right)_{j=1}^{m}, \left(g_1^{\frac{s^j\cdot Z(s)}{\delta}}\right)_{j=0}^{deg(Z)-2} \end{cases}$$

$$CRS_{G_1} = \Bigg\{ [6](13,15), [5](13,15), [4](13,15), ([2^0](13,15), [2^1](13,15)),$$
$$\left(\left[\frac{5A_0(2)+6A_0(2)+C_0(2)}{4}\right](13,15), \left[\frac{5A_1(2)+6A_1(2)+C_0(2)}{4}\right](13,15)\right)$$
$$\left(\left[\frac{5A_2(2)+6A_2(2)+C_2(2)}{3}\right](13,15), \left[\frac{5A_3(2)+6A_3(2)+C_3(2)}{3}\right](13,15),\right.$$
$$\left.\left[\frac{5A_4(2)+6A_4(2)+C_4(2)}{3}\right](13,15), \left[\frac{5A_5(2)+6A_5(2)+C_5(2)}{3}\right](13,15)\right),$$
$$\left(\frac{2^0T(2)}{3}(13,15)\right)\Bigg\}$$

The G_1 part of the CRS gives 12 points from *BLS6_6,*

$$\begin{aligned} CRS_{G_1} =& \{[6](13,15), [5](13,15), [4](13,15), ([1](13,15), [2](13,15)),\\ & ([0](13,15), [11](13,15)), ([2](13,15), [5](13,15), [10](13,15),\\ & [5](13,15)), [5](13,15)\}\\ CRS_{G_1} =& \{([0]g_1, [11]g_1), ([2]g_1, [5]g_1, [10]g_1, [5]g_1), [5]g_1\}\\ CRS_{G_1} =& \{[6]g_1, [5]g_1, [4]g_1, ([1]g_1, [2]g_1),\\ & ([0]g_1, [11]g_1), ([2]g_1, [5]g_1, [10]g_1, [5]g_1), [5]g_1\}\\ CRS_{G_1} =& \{(27,34), (26,34), (38,15), ((13,15), (33,34)),\\ & (Q, (33,9)), ((33,34), (26,34), (38,28), (27,9)), (26,34)\} \end{aligned}$$

The G_2 part of the CRS gives 5 points from *BLS6_6,*

$$\begin{aligned} CRS_{G_2} =& \big\{{g_2}^{\beta}, {g_2}^{\gamma}, {g_2}^{\delta}, \big({g_2}^{s^j}, ...\big)_{j=0}^{deg(Z)-1}\big\}\\ CRS_{G_2} =& \big\{[5](7v^2, 16v^3), [4](7v^2, 16v^3), [3](7v^2, 16v^3), ([1](7v^2, 16v^3), [2](7v^2, 16v^3))\big\}\\ CRS_{G_2} =& \big\{[5]g_2, [4]g_2), [3]g_2, ([1]g_2, [2]g_2)\big\}\\ CRS_{G_2} =& \big\{(16v^2, 28v^3), (37v^2, 27v^3), (42v^2, 16v^3), ((7v^2, 16v^3),\\ & (10v^2, 28v^3))\big\} \end{aligned}$$

Assume that the simulated trap door is removed. Given the assumed difficulty of the discrete logarithm issue in groups, it becomes impossible to determine the secret evaluation point, thus making the evaluation of polynomials at such a point impossible. Nevertheless, the polynomials evaluated at that particular location show a lower degree in relation to the coefficients of both generators, as compared to the degree of the desired polynomial.

Prover Stage:

Given a instance $I_1 =< 11 >$ and a witness vector $w = (w_1, w_2, w_3, \dots, w_m)$, in example $w =< 2, 3, 4, 6 >$. Here, it is necessary to refer to the definitions of witness and instance again. For this example,

Also, sample random value $r = 11$ and $t = 4$ from $\mathbb{F}_{13}$. Then, the proof is $\pi = (g_1^A, g_1^C, g_2^B)$.

$$g_1^w = \left(g_1^{\frac{\beta \cdot A_{n+1}(s) + \alpha \cdot B_{n+1}(s) + C_{n+1}(s)}{\delta}}\right)^{W_1} \dots \left(g_1^{\frac{\beta \cdot A_{n+m}(s) + \alpha \cdot B_{n+m}(s) + C_{n+m}(s)}{\delta}}\right)^{W_m}$$

$$g_1^A = g_1^a \cdot g_1^{A_0(s)} \cdot \left(g_1^{A_1(s)}\right)^{I_1} \dots \left(g_1^{A_n(s)}\right)^{I_n} \cdot \left(g_1^{A_{n+1}(s)}\right)^{W_1} \dots \left(g_1^{A_{n+m}(s)}\right)^{W_m} \cdot \left(g_1^{\delta}\right)^{r}$$

$$g_1^B = g_1^a \cdot g_1^{B_0(s)} \cdot \left(g_1^{B_1(s)}\right)^{I_1} \dots \left(g_1^{B_n(s)}\right)^{I_n} \cdot \left(g_1^{B_{n+1}(s)}\right)^{W_1} \dots \left(g_1^{B_{n+m}(s)}\right)^{W_m} \cdot \left(g_1^{\delta}\right)^{t}$$

$$g_2^B = g_1^a \cdot g_2^{B_0(s)} \cdot \left(g_2^{B_1(s)}\right)^{I_1} \dots \left(g_2^{B_n(s)}\right)^{I_n} \cdot \left(g_2^{B_{n+1}(s)}\right)^{W_1} \dots \left(g_2^{B_{n+m}(s)}\right)^{W_m} \cdot \left(g_2^{\delta}\right)^{t}$$

$$g_1^C = g_1^w \cdot g_1^{\frac{H(s) \cdot T(s)}{\delta}} \cdot \left(g_1^A\right)^t \cdot \left(g_1^B\right)^r \cdot \left(g_1\right)^{-rt}$$

Now, put the real number in formulas,

$$[W]g_1 = [W_1]g_1^{\frac{\beta \cdot A_2(\tau) + \alpha \cdot B_2(\tau) + C_2(\tau)}{3}} \oplus [W_2]g_1^{\frac{\beta \cdot A_3(\tau) + \alpha \cdot B_3(\tau) + C_3(\tau)}{3}} \oplus$$

$$[W_3]g_1^{\frac{\beta \cdot A_4(\tau) + \alpha \cdot B_4(\tau) + C_4(\tau)}{3}} \oplus [W_4]g_1^{\frac{\beta \cdot A_5(\tau) + \alpha \cdot B_5(\tau) + C_5(\tau)}{3}}$$

In order to calculate this particular point, it is critical that one keep in mind that the prover must not hold the simulation trapdoor. Consequently, the prover should remain ignorant of the specific values associated with the simulation trapdoor. To calculate the group element, the prover requires the CRS_G1.

To illustrate this point, let us examine the polynomials $A_2(x) = 6x + 10$ and $A_5(x) = 7x + 4$, which are part of the QAP under consideration. In order to assess the polynomials in the exponent of g_1 and g_2 at the undisclosed point τ, while remaining unaware of the specific value of τ (which, in this instance, is 2), the CRS and equation may be used.

$$\begin{aligned} CRS_{G_1} = \{&(27, 34), (26, 34), (38, 15), ((13, 15), (33, 34)), \\ &(Q, (33, 9)), ((33, 34), (26, 34), (38, 28), (27, 9)), (26, 34)\} \end{aligned}$$

$$CRS_{G_2} = \left\{(16v^2, 28v^3), (37v^2, 27v^3), (42v^2, 16v^3), ((7v^2, 16v^3), (10v^2, 28v^3))\right\}$$

The colors show where the CRS parameters are used.

$$\begin{aligned} [W]g_1 &= [2](33, 34) \oplus [3](26, 34) \oplus [4](38, 28) \oplus [6](27, 9) \oplus \\ &= [2 \cdot 2](13, 15) \oplus [3 \cdot 5](13, 15) \oplus [4 \cdot 10](13, 15) \oplus [6 \cdot 7](13, 15) \\ &= [4 + 15 + 40 + 42](13, 15) = [10](13, 15) = [10]g_1 \\ &= (38, 28) \end{aligned}$$

Given that the values of α, δ, and τ are currently unknown, it is recommended to search for the values $[\alpha]g1$ and $[\delta]g1$ inside the CRS_G1.

$$\begin{aligned} [A]g_1 = {} & [\alpha]g_1 \oplus [A_0(\tau)]g_1 \oplus [I_1][A_1(\tau)]g_1 \oplus [W_1][A_2(\tau)]g_1 \oplus \\ & \oplus [W_2][A_3(\tau)]g_1 \oplus [W_3][A_4(\tau)]g_1 \oplus [W_4][A_5(\tau)]g_1 \\ & \oplus [r][\delta]g_1 \\ = {} & [6]g_1 \oplus [2][A_2(\tau)]g_1 \oplus [6][A_5(\tau)]g_1 \oplus [11][3]g_1 \end{aligned}$$

Find $[A_2(\tau)]g_1$ and $[A_5(\tau)]g_1$

$$\begin{aligned} [A_2(\tau)]g1 &= [6 \cdot \tau^1 \oplus 10 \cdot \tau^0]g1 \\ &= [6](33, 34) \oplus [10](13, 15) \\ &= [6 \cdot 2](13, 15) \oplus [10](13, 15) = [9](13, 15) \\ &= (35, 15) \end{aligned}$$

$$
\begin{aligned}
[A_5(\tau)]g1 &= [7 \cdot \tau^1 \oplus 4 \cdot \tau^0]g1 \\
&= [7](33, 34) \oplus [4](13, 15) \\
&= [7 \cdot 2](13, 15) \oplus [4](13, 15) = [5](13, 15) \\
&= (26, 34)
\end{aligned}
$$

$$
\begin{aligned}
[A]g_1 &= (27, 34) \oplus [2](35, 15) \oplus [6](26, 34) \oplus [11](38, 15) \\
&= [6](13, 15) \oplus [2 \cdot 9](13, 15) \oplus [6 \cdot 5](13, 15) \oplus [11 \cdot 3](13, 15) \\
&= [6 + 2 \cdot 9 + 6 \cdot 5 + 11 \cdot 3](13, 15) = [9](13, 15) \\
&= (35, 15)
\end{aligned}
$$

According to QAP, $B_3 = A_2$ and $B_4 = A_5$

$$
\begin{aligned}
[B]g_1 &= [\beta]g_1 \oplus [B_0(\tau)]g_1 \oplus [I_1][B_1(\tau)]g_1 \oplus [W_1][B_2(\tau)]g_1 \oplus \\
&\oplus [W_2][B_3(\tau)]g_1 \oplus [W_3][B_4(\tau)]g_1 \oplus [W_4][B_5(\tau)]g_1 \oplus [t][\delta]g_1 \\
&= (26, 34) \oplus [3](35, 15) \oplus [4](26, 34) \oplus [4](38, 15) \\
&= [5](13, 15) \oplus [3 \cdot 9](13, 15) \oplus [4 \cdot 5](13, 15) \oplus [4 \cdot 3](13, 15) \\
&= [5 + 3 \cdot 9 + 4 \cdot 5 + 4 \cdot 3](13, 15) = [12](13, 15) \\
&= (13, 28)
\end{aligned}
$$

$$
\begin{aligned}
[B]g_2 &= [\beta]g_2 \oplus [B_0(\tau)]g_2 \oplus [I_1][B_1(\tau)]g_2 \oplus [W_1][B_2(\tau)]g_2 \oplus \\
&\oplus [W_2][B_3(\tau)]g_2 \oplus [W_3][B_4(\tau)]g_2 \oplus [W_4][B_5(\tau)]g_2 \oplus [t][\delta]g_2 \\
&= [5]g_2 \oplus [3][B_3(\tau)]g_2 \oplus [4][B_4(\tau)]g_2 \oplus [4][3]g_2
\end{aligned}
$$

Find $[B_3(\tau)]g_2$ and $[B_4(\tau)]g_2$

$$
\begin{aligned}
[B_3(\tau)]g2 &= [6 \cdot \tau^1 \oplus 10 \cdot \tau^0]g1 \\
&= [6](10v^2, 28v^3) \oplus [10](7v^2, 16v^3) \\
&= [6 \cdot 2](7v^2, 16v^3) \oplus [10](7v^2, 16v^3) = [9](7v^2, 16v3) \\
&= (37v^2, 16v^3)
\end{aligned}
$$

$$
\begin{aligned}
[B_4(\tau)]g2 &= [7 \cdot \tau^1 \oplus 4 \cdot \tau^0]g1 \\
&= [7](10v^2, 28v^3) \oplus [4](7v^2, 16v^3) \\
&= [7 \cdot 2](7v^2, 16v^3) \oplus [4](7v^2, 16v^3) = [5](7v^2, 16v3) \\
&= (16v^2, 28v^3)
\end{aligned}
$$

$$
\begin{aligned}
[B]g_2 &= (16v^2, 28v^3) \oplus [3](37v^2, 16v^3) \oplus [4](16v^2, 28v^3) \oplus [4](42v^2, 16v^3) \\
&= [5](7v^2, 16v^3) \oplus [3 \cdot 9](7v^2, 16v^3) \oplus [4 \cdot 5](7v^2, 16v^3) \oplus [4 \cdot 3](7v^2, 16v^3) \\
&= [5 + 3 \cdot 9 + 4 \cdot 5 + 4 \cdot 3](7v^2, 16v^3) = [12](7v^2 + 16v^3) \\
&= (7v^2, 27v^3)
\end{aligned}
$$

$$
\begin{aligned}
[C]g_1 &= [W]g_1 \oplus [\frac{H(s)Z(s)}{\delta}]g_1 \oplus [t][A]g_1 \oplus [r][B]g_1 \oplus [-rt][\delta]g_1 \\
&= (38, 28) \oplus (26, 34) \oplus [4](35, 15) \oplus [11](13, 28) \oplus [-11 \cdot 4](38, 15) \\
&= [10](13, 15) \oplus [5](13, 15) \oplus [4 \cdot 9](13, 15) \oplus [11 \cdot 12](13, 15) \oplus [-11 \cdot 4 \cdot 3](13, 15) \\
&= [10 + 5 + 4 \cdot 9 + 11 \cdot 12 - 11 \cdot 4 \cdot 3](13, 15) = [12](13, 15) \\
&= (13, 28)
\end{aligned}
$$

Finally,

$$\pi = \big((35, 15), (13, 28), (7v^2, 27v^3)\big)$$

Verification Stage:

Know π, and verification of proof is very easy. The pairing is verified,

$$e({g_1}^A, {g_2}^B) = e({g_1}^\alpha, {g_2}^\beta) \cdot e({g_1}^I, {g_2}^\gamma) \cdot e({g_1}^C, {g_2}^\delta)$$

Remember that,

$$\pi = ({g_1}^A, {g_1}^C, {g_2}^B)$$

Also,

$$g_1{}^I = \left(g_1{}^{\frac{\beta\cdot A_0(\tau)+\alpha\cdot B_0(\tau)+C_0(\tau)}{\gamma}}\right)\cdot\left(g_1{}^{\frac{\beta\cdot A_1(\tau)+\alpha\cdot B_1(\tau)+C_1(\tau)}{\gamma}}\right)^{I_1}\cdots\left(g_n{}^{\frac{\beta\cdot A_n(\tau)+\alpha\cdot B_n(\tau)+C_n(\tau)}{\gamma}}\right)^{I_n}$$

$$\begin{aligned} g_1{}^I &= \mathcal{O}\oplus[11](33,9) = [11\cdot 11](13,15) = [4](13,15) \\ &= (35,28) \end{aligned}$$

The Weil Pairing Rule refers that,

$$e([a]g_1,[b]g_2) = e(g_1,g_2)^{a*b}$$

So,

$$\begin{aligned} e([A]g1,[B]g2) &= e((35,15),(7v^2,27v^3)) = e([9](13,15),[12](7v^2,16v^3)) \\ &= e((13,15),(7v^2,16v^3))^{9\cdot 12} \\ &= e((13,15),(7v^2,16v^3))^{108} \\ e([\alpha]g1,[\beta]g2) &= e((27,34),(16v^2,28v^3)) = e([6](13,15),[5](7v^2,16v^3)) \\ &= e((13,15),(7v^2,16v^3))^{6\cdot 5} \\ &= e((13,15),(7v^2,16v^3))^{30} \\ e([I]g1,[\gamma]g2) &= e((35,28),(37v^2,27v^3)) = e([4](13,15),[4](7v^2,16v^3)) \\ &= e((13,15),(7v^2,16v^3))^{4\cdot 4} \\ &= e((13,15),(7v^2,16v^3))^{16} \\ e([C]g1,[\delta]g2) &= e((13,28),(42v^2,16v^3)) = e([12](13,15),[3](7v^2,16v^3)) \\ &= e((13,15),(7v^2,16v^3))^{12\cdot 3} \\ &= e((13,15),(7v^2,16v^3))^{36} \end{aligned}$$

Finally,

$$e(g_1,g_2)^{9*12} = e(g_1,g_2)^{6*5}\cdot e(g_1,g_2)^{4*4}\cdot e(g_1,g_2)^{12*3}$$

The Weil pairing may be characterized as a finite cyclic group with an order of 13. Consequently, the process of exponentiation is performed within the structure of modular 13 arithmetic.

$$e(g_1,g_2)^4 = e(g_1,g_2)^4$$

The verification part is finished. It is evident that the left and right sides of the equation show similar characteristics, so indicating that the verification process acknowledges the validity of the zk-SNARK and encourages the verifier to generate an acceptance response.

CHAPTER 3

MACHINE LEARNING

3.1 Definition of Machine Learning

Machine learning (ML) is a methodology designed to optimize the computational capabilities of computers in processing and managing data. There are instances in which it is impossible to accurately evaluate the knowledge obtained from data based on its apparent meaning. In such situations, ML becomes relevant. In the contemporary context, in which many kinds of datasets are easily accessible, there has been a consistent increase in the need for machine learning. This technology has been widely used in diverse industries to extract essential information and is naturally built to acquire knowledge from patterns inside data.

Numerous studies have been conducted to investigate methods of facilitating autonomous learning in computers without the need for explicit programming. Various approaches have been investigated by mathematicians and programmers in order to tackle this difficulty, namely within the realm of large datasets. In order to overcome the problems presented by the intricate nature of data, machine learning utilizes a diverse range of algorithms. Data scientists emphasize that there is no generally optimal algorithm that can effectively address all issues. The selection of an algorithm must be customized to suit the individual situation under consideration.

3.2 Supervised Learning and Unsupervised Learning

Supervised learning is very common in classification difficulties since the goal is typically to get the computer to learn a categorization system that we have created. Classification learning, in a larger sense, is appropriate for any circumstance where it is profitable to infer a classification and if doing so is straightforward. As long as the inputs are available, this model is not necessary, but it is impossible to draw any conclusions about the outputs if some of the input values are missing.

When learning without supervision its name is unsupervised learning, it is believed that all observations are the result of latent variables, placing them at the very end of the causal chain. Unsupervised learning, when the machine is supposed to figure out things on its own rather than being given instructions, appears to be far more difficult. Actually, there are two methods for unsupervised learning. The first strategy is to instruct the agent by employing a reward system to denote success rather than by providing explicit categorizations. This strategy generalizes well to the actual world, where agents may get rewards for doing some behaviors and penalties for others. Clustering is a second kind of unsupervised learning. The objective of this kind of learning is to merely identify commonalities within the training data rather than maximizing a utility function. It is frequently assumed that the found clusters would rather closely fit an intuitive categorisation. For instance, grouping people based on demographics may cause the rich to be grouped in one category and the destitute in another. Despite the fact that the algorithm won't have names to give these clusters, it may still create them and utilize those clusters to place fresh instances into either one of the clusters or the other.

3.3 Process of Developing Model

Because the thesis focuses on the supervised learning method, this section addresses the process of applying supervised machine learning to a real-world problem. According to The study "Types of machine learning algorithms" (Ayodele, 2010) [2], this thesis includes details of process. The dataset must be gathered initially. The most informative fields (attributes, characteristics) might be suggested if the necessary expert is on hand. If not, the most straightforward approach is "brute-force," which entails measuring all characteristics in the hopes that the appropriate ones may be identified.

Preparing and processing the data is the second phase. Researchers can manage missing data in a number of ways, depending on the circumstances.

The act of discovering and eliminating as many unnecessary and redundant features as you can is known as feature subset selection. As a result, the data's dimensions are reduced, making it possible for data mining algorithms to work more quickly and efficiently.

The development of clearer and more precise classifiers might be facilitated by these recently developed characteristics. Additionally, the identification of significant traits helps people comprehend the acquired topic and the created classifier more clearly. The instances from which the agent tries to learn are these inputs, also known as the "training set". It's not always the best idea to thoroughly understand the training set, though. For instance, if I attempted to teach you exclusive or but all the combinations I gave you were one true and one false, never both false or both true, you may learn that the answer is always true.

Not all training sets contain appropriately categorized inputs, as you might expect. This might cause issues if the algorithm is strong enough to remember even the ostensibly "special cases" that don't meet the more general rules.

Finding algorithms that are both potent enough to learn complicated functions and reliable enough to generate findings that can be generalized is difficult because this may also result in over fitting.

The final state is then achieved after the model has been updated repeatedly using the test set.

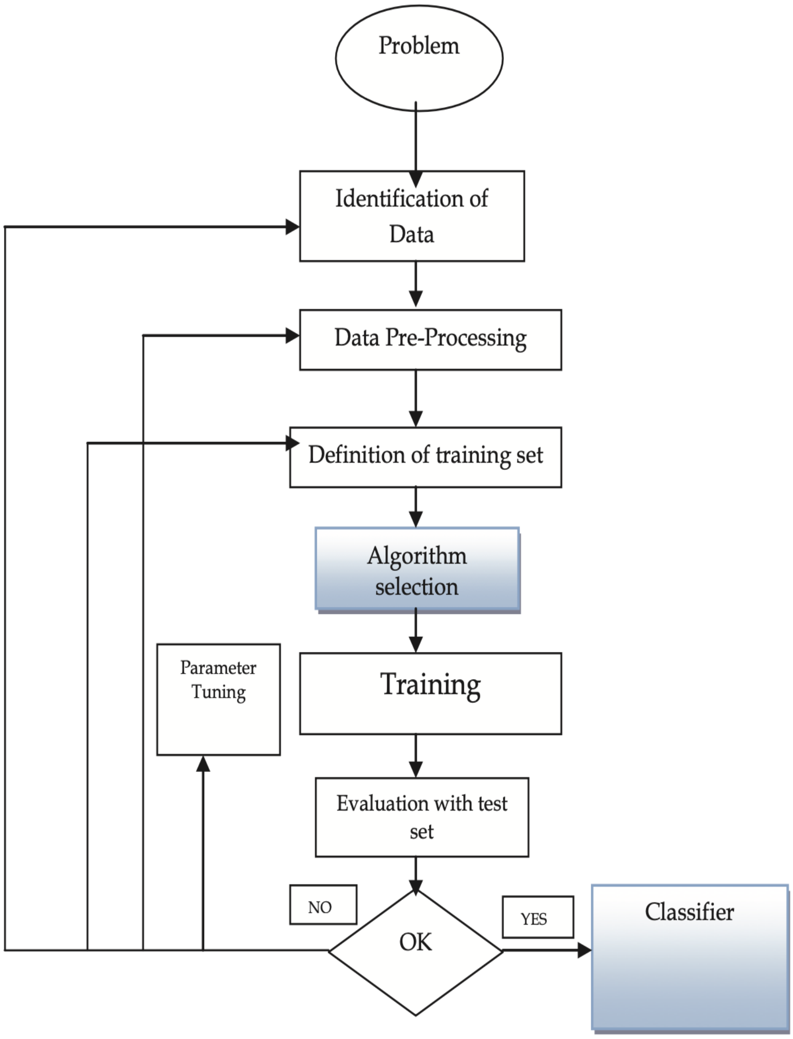


Figure 3.1: Process of Machine Learning [2]

3.4 Importance and relevance of privacy in ML

The following section explains several reasons that may give rise to security problems.

Private individual information

Any active user of information services leaves a lot of personal data in the open network that are either required for the fulfillment of their requests or due to legal requirements, such as name and surname, passport information, information about where they are located, the items they have purchased, the specifics of their financial transactions, etc. Therefore, the presence of personal data in the network opens the door to the possibility of its collection, analysis, and use for both beneficial and detrimental ends. Although most nations have laws requiring the protection of personal data, these laws often can't stop the exposure of personal data and its misuse. Additionally, consumers frequently "tick" the appropriate boxes without properly reading the privacy regulations for information services (or without reading them at all). Because of this, a normal user eventually forgets where and what personal information he left behind.

The training sample may contain some confidential, in particular, private data. Owners of this data may be interested in having being used them to train a model, but do not want to lose control of it. The model owner may also not want the model parameters to be known to others, including those who provide data to train the model.

When sensitive data is utilized to train models in conventional settings, privacy is a major problem in the area of machine learning. If the models were trained on sensitive data, sharing machine learning models, especially pre-trained models, may give rise to privacy problems. Such algorithms could unintentionally leak private data from training sets on specific people if they are released. When sensitive information is accidentally included during model training, it's referred to as data leakage in machine learning. A model might mistakenly learn to generate predictions based on sensitive information if it is trained on data that contains information that it shouldn't have access to.

When an attacker can extrapolate private information about a person from a trained model's predictions, this is known as an inference assault. Even if the model does not directly access private data during training, its predictions may nevertheless include sensitive information.In model inversion attacks, an adversary looks at the results of a machine learning model to try to recreate personal data about individuals. This can be particularly troublesome if the output of the model is utilized to make decisions in delicate areas.

Identification of Risks and Manipulation of the Model

The examination and comprehension of a model's decision-making process may be facilitated by an attacker via the access to the model's parameters. This facilitates comprehension about the rationale behind the model's decision-making process, the specific attributes it prioritizes, and the data samples to which it reacts with more precision. Based on the provided data, potential assailants possess the ability to discern weaknesses within the model and then engage in manipulative actions to exploit those weaknesses. As an example, one might strategically focus on certain areas in which the model exhibits vulnerabilities, hence enabling the circumvention of security mechanisms or the generation of deceptive data. An instance of unauthorized access to the parameters of the image processing model might enable an adversary to generate counterfeit pictures or execute changes with the intention of evading recognition systems.

Competition-related Concerns

The interception or disclosure of model parameters raises substantial security issues related to protecting competitive techniques and advances in the field of artificial intelligence. This facilitates competitors in gaining awareness of the fundamental principles of powerful AI models and using this knowledge for their own advantage. Performing a competitor analysis involves the divulgence of important information, such as the structural configuration of the model, the learning algorithms applied, the data processing techniques utilized, and the adjustments made to hyperparameters. This facilitates competitors to accelerate the advancement of their own models, get more effective results, and attain a competitive advantage.

Moreover, the disclosure of model weights have the capacity to compromise private knowledge and trade secrets. The unauthorized disclosure of sensitive information or proprietary knowledge inside an organization may lead to enormous losses in terms of time and financial resources. This has the potential to undermine extensive research and development efforts and large monetary investments. Moreover, the act of confiscating weights might possibly suggest the exploitation of the model. The use of the model by persons with malicious intentions to produce misleading or harmful content has the capacity to mislead the broader populace, undermine security measures, and lead to substantial social and economic consequences.

Therefore, it is crucial to prioritize the maintenance of secrecy and privacy regarding model weights in the field of artificial intelligence and deep learning.

3.5 Zero Knowledge Machine Learning

ZKML has the potential to facilitate several novel applications in practical settings due to its robust assurances about privacy and integrity. So, the customers are not granted access to the proprietary machine learning models owned by the firms. Consequently, they must rely on the assurance that the forecasts are being accurately calculated as stipulated.

By using ZKML, the service provider is able to demonstrate the superior quality and precision of the models, as well as verify that the forecasts are derived from these same models. In some instances, the validation of claims on the high accuracy achieved by certain machine learning models or algorithms poses a significant challenge. ZKML offers a partial resolution to this problem. By including a ZKP to verify the asserted precision, it demonstrates the existence of a machine learning model acknowledged by the proprietor, as supported by the ZKP's knowledge soundness.

In an alternative context, ZKML may also facilitate the prediction of public models on confidential datasets.

It may also be used to demonstrate the existence of adversarial instances for a publicly accessible model, whereby two inputs exhibit proximity to each other but are assigned to distinct categories.

3.6 Neural Networks

Neural networks are a collection of algorithms that are roughly based on the human brain and have the goal of identifying patterns. They analyze sensory data by categorizing or grouping raw input using machine perception. They identify numerical patterns contained in vectors, into which all real-world data, whether pictures, music, text, or time series, must be transformed.

Neural networks are used to cluster and categorize data. Consider them a grouping and classification layer above the data you collect and manage, and also aid in the grouping of unlabeled data based on similarities between example inputs.

Network designs are classified into two types based on the kind of connections between neurons: *"feed-forward neural networks"* and *"recurrent neural networks."* The network is referred regarded as a "feed-forward neural network" if there is no "feedback" from the neurons' outputs to the inputs throughout the network. Otherwise, if such feedback occurs, i.e. a synaptic link from the outputs to the inputs , the network is referred to as a "recurrent neural network." Neural networks are often organized in "layers." Feed-forward neural networks are classified as "single layer" or "multi-layer" based on the number of layers.

Neuron: The model characterizes a neuron as a binary processing unit. The firing state of a neuron is determined by its output signal, which is regulated by a threshold logic or *activation function*.

The model neuron, much to its real counterpart, has a multitude of inputs and a single output. Each input to the neuron is associated with a *weight*. In order to determine whether a neuron will generate an output signal, it is necessary to compute the summation of the products obtained by multiplying each input with its corresponding synaptic weight. The activation function is then computed using the summation value obtained.

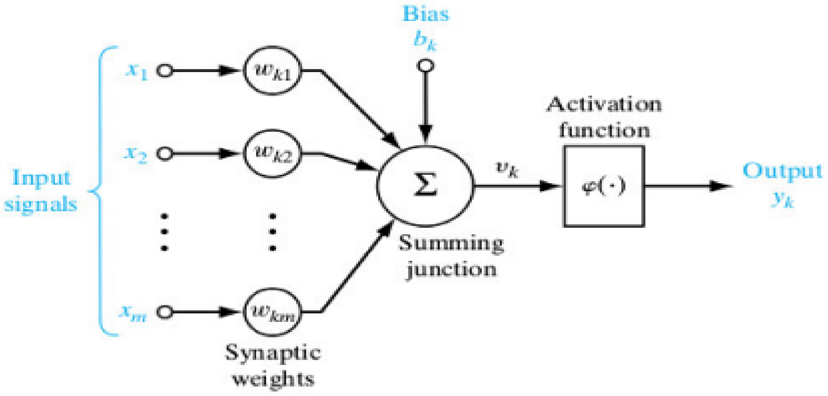


Figure 3.2: Neuron [28]

3.6.1 Neural Networks Structure with Python Code

The MNIST [18] handwritten digit classification issue is a well recognized dataset often used in the field of computer vision. In addition, The abbreviation MNIST represents the Modified National Institute of Standards and Technology dataset.

While the dataset has been mostly solved, it may still serve as a valuable resource for learning and honing the skills required to create, assess, and use neural networks for the purpose of picture categorization starting from the ground up. This encompasses the process of constructing a reliable test framework to evaluate the model's performance, investigating potential enhancements to the model, and implementing mechanisms to store and subsequently use the model for making predictions on novel data.

This part aims to elucidate the process of constructing a neural network for the purpose of classifying handwritten digits, starting from the very beginning. A neural network model will be set up as shown below.

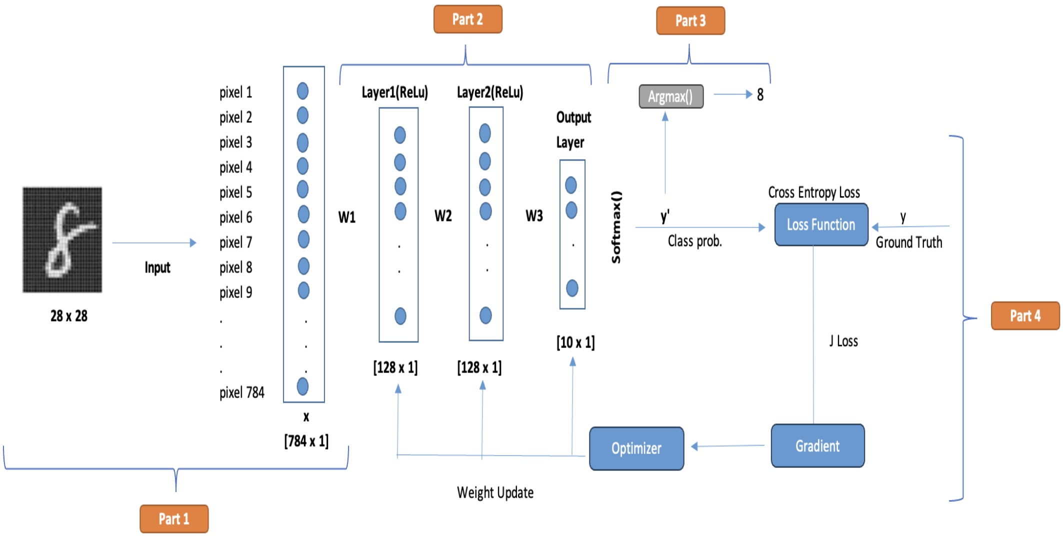


Figure 3.3: Neural Netwrok Structure

Part1: Dataset Preprocessing

The collection consists of 60,000 grayscale photos of handwritten single digits ranging from 0 to 9. Each image is a tiny square with dimensions of 28×28 pixels. The objective is to categorize a provided picture of a handwritten numeral into one of ten distinct categories that correspond to integer values ranging from 0 to 9, inclusively.

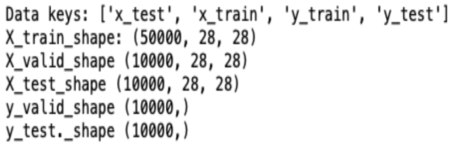
Python code 1:

```
# load .npz file
data = np.load('mnist.npz')
X_train_all, y_train_all = data['x_train'],data['y_train']
X_test,y_test = data['x_test'],data['y_test']

X_valid = X_test[:10000]
X_train = X_train_all[10000:]
y_valid = y_test[:10000]
y_train = y_train_all[10000:]

data.close()
```

Output:



```
Data keys: ['x_test', 'x_train', 'y_train', 'y_test']
X_train_shape: (50000, 28, 28)
X_valid_shape (10000, 28, 28)
X_test_shape (10000, 28, 28)
y_valid_shape (10000,)
y_test._shape (10000,)
```

Figure 3.4: Output of Python code 1

- Training dataset: Subset of data that is used to train a model.
- Validation dataset: Subset of data that is used to impartially assess the performance of a model that has been trained on a separate dataset, with the purpose of fine-tuning the model's hyperparameters.
- Test dataset: Subset of data that is used to impartially assess the performance of a trained model, which has been fitted on the training dataset.

Additionally, a visual representation of the first three photos within the dataset is generated, showcasing the inherent handwriting characteristics of the images that are to be categorized.

Python code 2:

```
# 28 x 28 picture
plt.figure(figsize=(7,3))
for i in range(3):
    plt.subplot(1, 3, i + 1)
    plt.axis(True)
    plt.imshow(X_train[i], cmap='gray')
    plt.subplots_adjust(wspace=0.2, hspace=0.2)
```

Output:

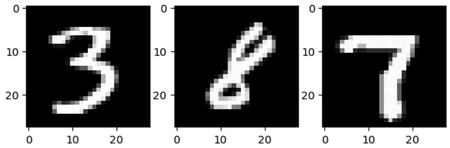


Figure 3.5: Output of Python code 2

The collection contains images with pixel values that are either black and white or unsigned numbers within the range of 0 to 255. A recommended first step in the process is to normalize the pixel values of grayscale pictures, such as by rescaling them to the range of [0,1]. The first step in this process is transforming the data type from unsigned integers to floating-point numbers, followed by dividing the pixel values by the maximum value.

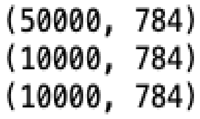
Python code 3:

```
X_train = X_train.reshape((X_train.shape[0],28 * 28))
X_train = X_train.astype("float32") / 255

X_test = X_test.reshape((X_test.shape[0],28 * 28))
X_test = X_test.astype("float32") / 255

X_valid = X_valid.reshape((X_valid.shape[0],28 * 28))
X_valid = X_valid.astype("float32") / 255
```

Output:



```
(50000, 784)
(10000, 784)
(10000, 784)
```

Figure 3.6: Output of Python code 3

Next, use a *one-hot encoding technique* to represent the class attribute of each sample. This process involves converting the integer representation of the class into a binary vector consisting of ten elements. In this vector, a value of 1 is assigned to the index corresponding to the class value, while all other indices are assigned a value of 0. This may be accomplished using the $to\_categorical()$ function.

Python code 4:

```
y_train = to_categorical(y_train)
y_valid = to_categorical(y_valid)
y_test = to_categorical(y_test)
```

Part2: Layers

Neural networks encode information or abstract patterns in parameters known as *weights* and *biases*. Neural networks are comprised of *layers*, with each layer including *nodes*. Also, the weights and bias are often regarded as fundamental components inside a neural network. In the process of transmitting inputs between neurons, the inputs undergo a multiplication with corresponding weights, followed by their summation with a bias term. This combined value is then fed into an activation function.

This is an illustrative neural network architecture comprising of an input layer, two hidden layers, and an output layer. During training a neural network using the training set, the network is first assigned a group of weights. The first step of neural computation involves the calculation of the weighted sum of the input signals. The multiplication operation is performed between the inputs and weights, resulting in the computation of a weighted sum. Following that, a constant bias is introduced into the weighted total. Ultimately, the calculated value is inputted into the activation function, which produces an output.

$$f_i(x) = w_i x + b_i$$

Activation function - ReLU

Model inputs are the primary components of a neural network's design. These inputs are processed and transformed by an activation or threshold function to activate the unit, producing an output. Here, the ReLu function will be emphasized since it will be utilized.

Rectified linear unit, often called ReLU, is a non-linear activation function widely used in neural networks. Utilizing the Rectified Linear Unit (ReLU) function prevents simultaneous stimulation of all neurons. This implies that a neuron will become inactive once the result of the linear transformation equals zero. It may be described as follows using mathematics:

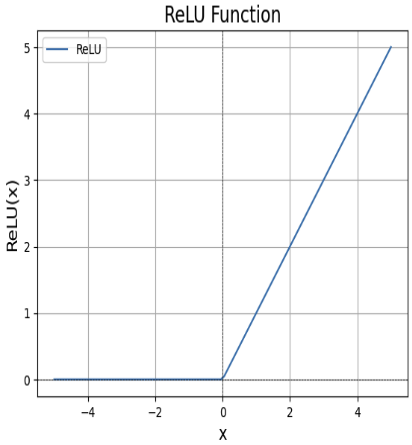


Figure 3.7: ReLU Graph

ReLU function is considered more efficient compared to other activation functions due to its selective activation of neurons. Unlike other functions, ReLU activates a specific number of neurons at any one moment, rather than activating all neurons simultaneously.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Part3: Class Probability

Softmax activation function

The Softmax activation function calculates the relative probabilities and it returns the probability of each class. Here, the Z represents the values from the neurons of the output layer. The exponential acts as the non-linear function. Later these values are divided by the sum of exponential values in order to normalize and then convert them into probabilities.

$$y' = softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_i)}$$

Argmax Function

The argmax function is used to determine the argument or arguments (arg) that correspond to the greatest (max) value obtained from the target function.

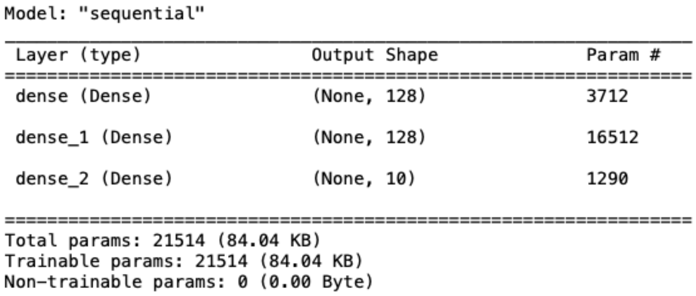
Python code 5:

```
model = tf.keras.Sequential()

model.add(Dense(128,activation=’relu’,input_shape=(X_train.shape[1],)))
model.add(Dense(128,activation=’relu’, ))
model.add(Dense(10,activation =’softmax’))

model.summary()
```

Output:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               3712

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 21514 (84.04 KB)
Trainable params: 21514 (84.04 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 3.8: Output of Python code 5

Part4: Loss Function, Gradient

Cross Entropy Loss Function

Within the framework of an optimization procedure, the objective function is used to assess a prospective solution, often known as a set of weights.

The objective function may be optimized by either maximizing or minimizing it, indicating the pursuit of a candidate solution with the greatest or lowest score, respectively.

In the context of neural networks, the objective is often to reduce the error. Therefore, the objective function is often denoted as a loss function, and the computed value given by the loss function is commonly referred to as "loss."

It is the Multi-class classification problem, so here Cross-Entropy Loss is used as the loss function.

The term "cross-entropy loss" is often denoted as "cross-entropy" or "logarithmic loss" in academic literature.

The projected probabilities are compared to the actual class output result, which may be either 0 or 1. A score is then generated, taking into account the deviation of the probability from the anticipated value. The penalty exhibits a logarithmic relationship, whereby little discrepancies (e.g., 0.1 or 0.2) are associated with relatively low scores, while substantial discrepancies (e.g., 0.9 or 1.0) result in much higher scores.A model that accurately predicts probabilities with a cross entropy value of 0.

$$\mathcal{L} = -\sum_{k} y_k log(y'_k)$$

Gradient

In a broad context, the concept of gradient pertains to the measure of the slope of an equation. Gradients, on the other hand, refer to partial derivatives, which serve to characterize the alteration seen in the loss function in relation to slight modifications in the function's parameters. This little modification in the loss functions provides insights into the subsequent approach for mitigating the output of the loss function.

In this model, the RMSprop gradient algorithm is used. RMSprop operates by computing the gradient of the loss function relative to the parameters of the model and thereafter adjusting the parameters in the direction opposite to the gradient in order to minimize the loss. A notable characteristic of this approach is the use of a moving average of the squared gradients to adjust the learning rate. This intervention contributes to the stabilization of the learning process.

Python code 6:

```
model.compile(optimizer="rmsprop",
              loss="categorical_crossentropy",
              metrics=["accuracy"])
```

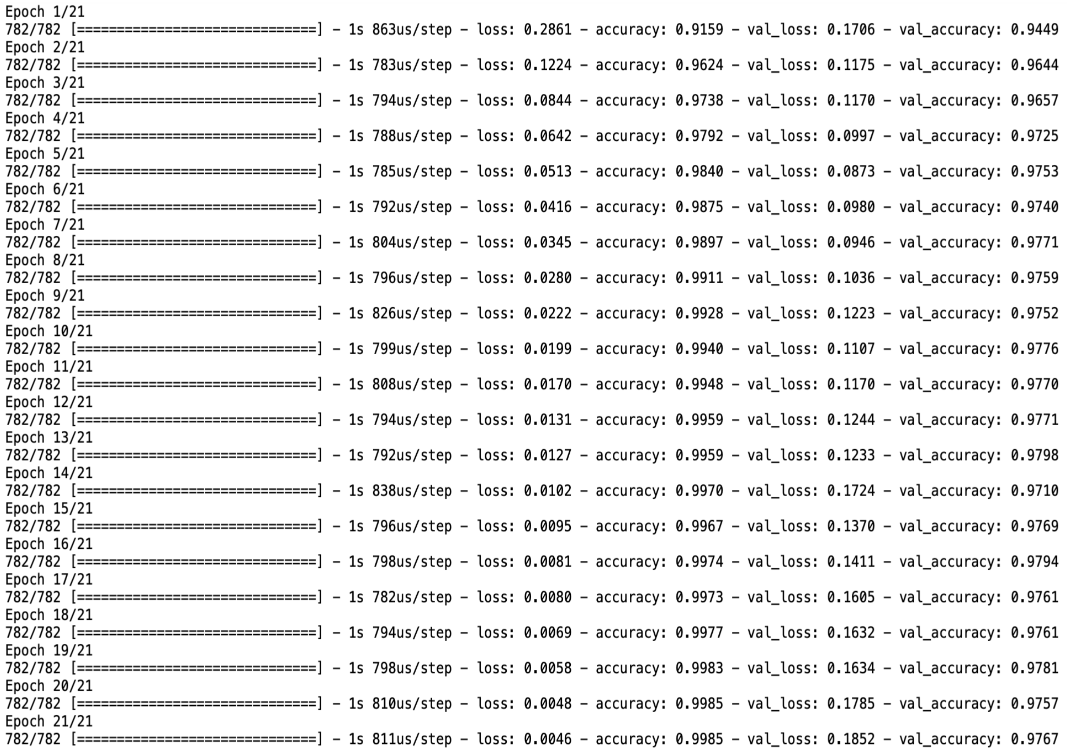
Lastly, the model is trained with the help of the code below, the meanings of the terms here are as follows.

- **Epoch:** The frequency at which the algorithm iterates over the whole training dataset.
- **Batch:** It denotes the number of samples to be taken to for updating the model parameters.

Python code 7:

```
training_results = model.fit(X_train,
                             y_train,
                             epochs=21,
                             batch_size=64,
                             validation_data=(X_valid, y_valid))
```

Output:



```
Epoch 1/21
782/782 [==============================] - 1s 863us/step - loss: 0.2861 - accuracy: 0.9159 - val_loss: 0.1706 - val_accuracy: 0.9449
Epoch 2/21
782/782 [==============================] - 1s 783us/step - loss: 0.1224 - accuracy: 0.9624 - val_loss: 0.1175 - val_accuracy: 0.9644
Epoch 3/21
782/782 [==============================] - 1s 794us/step - loss: 0.0844 - accuracy: 0.9738 - val_loss: 0.1170 - val_accuracy: 0.9657
Epoch 4/21
782/782 [==============================] - 1s 788us/step - loss: 0.0642 - accuracy: 0.9792 - val_loss: 0.0997 - val_accuracy: 0.9725
Epoch 5/21
782/782 [==============================] - 1s 785us/step - loss: 0.0513 - accuracy: 0.9840 - val_loss: 0.0873 - val_accuracy: 0.9753
Epoch 6/21
782/782 [==============================] - 1s 792us/step - loss: 0.0416 - accuracy: 0.9875 - val_loss: 0.0980 - val_accuracy: 0.9740
Epoch 7/21
782/782 [==============================] - 1s 804us/step - loss: 0.0345 - accuracy: 0.9897 - val_loss: 0.0946 - val_accuracy: 0.9771
Epoch 8/21
782/782 [==============================] - 1s 796us/step - loss: 0.0280 - accuracy: 0.9911 - val_loss: 0.1036 - val_accuracy: 0.9759
Epoch 9/21
782/782 [==============================] - 1s 826us/step - loss: 0.0222 - accuracy: 0.9928 - val_loss: 0.1223 - val_accuracy: 0.9752
Epoch 10/21
782/782 [==============================] - 1s 799us/step - loss: 0.0199 - accuracy: 0.9940 - val_loss: 0.1107 - val_accuracy: 0.9776
Epoch 11/21
782/782 [==============================] - 1s 808us/step - loss: 0.0170 - accuracy: 0.9948 - val_loss: 0.1170 - val_accuracy: 0.9770
Epoch 12/21
782/782 [==============================] - 1s 794us/step - loss: 0.0131 - accuracy: 0.9959 - val_loss: 0.1244 - val_accuracy: 0.9771
Epoch 13/21
782/782 [==============================] - 1s 792us/step - loss: 0.0127 - accuracy: 0.9959 - val_loss: 0.1233 - val_accuracy: 0.9798
Epoch 14/21
782/782 [==============================] - 1s 838us/step - loss: 0.0102 - accuracy: 0.9970 - val_loss: 0.1724 - val_accuracy: 0.9710
Epoch 15/21
782/782 [==============================] - 1s 796us/step - loss: 0.0095 - accuracy: 0.9967 - val_loss: 0.1370 - val_accuracy: 0.9769
Epoch 16/21
782/782 [==============================] - 1s 798us/step - loss: 0.0081 - accuracy: 0.9974 - val_loss: 0.1411 - val_accuracy: 0.9794
Epoch 17/21
782/782 [==============================] - 1s 782us/step - loss: 0.0080 - accuracy: 0.9973 - val_loss: 0.1605 - val_accuracy: 0.9761
Epoch 18/21
782/782 [==============================] - 1s 794us/step - loss: 0.0069 - accuracy: 0.9977 - val_loss: 0.1632 - val_accuracy: 0.9761
Epoch 19/21
782/782 [==============================] - 1s 798us/step - loss: 0.0058 - accuracy: 0.9983 - val_loss: 0.1634 - val_accuracy: 0.9781
Epoch 20/21
782/782 [==============================] - 1s 810us/step - loss: 0.0048 - accuracy: 0.9985 - val_loss: 0.1785 - val_accuracy: 0.9757
Epoch 21/21
782/782 [==============================] - 1s 811us/step - loss: 0.0046 - accuracy: 0.9985 - val_loss: 0.1852 - val_accuracy: 0.9767
```

Figure 3.9: Output of Python code 7

In this particular scenario, it is evident that the model exhibits a satisfactory level of conformity, as shown by the convergence of the train and test learning curves. There is no discernible indication of either over-fitting or under-fitting.

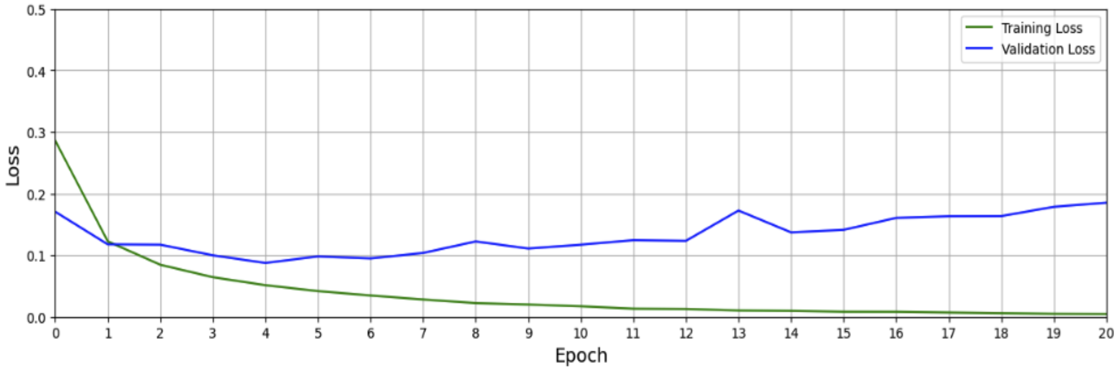


Figure 3.10: Train Loss and Validation Loss for each epoch

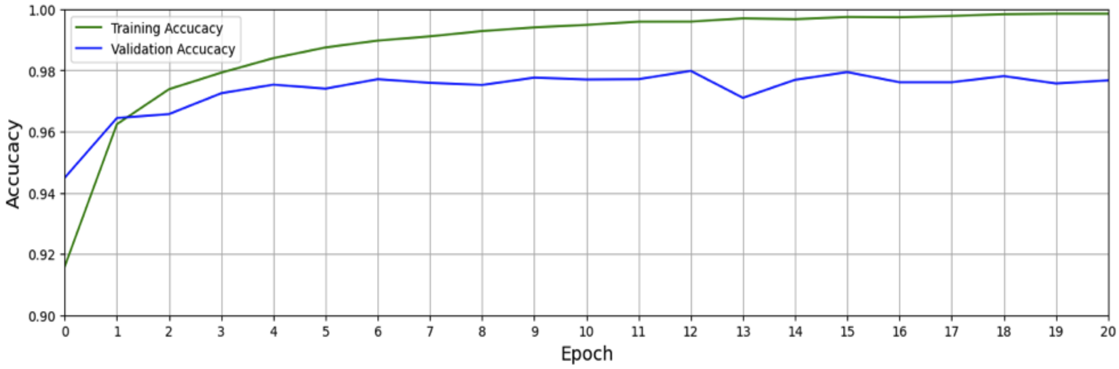
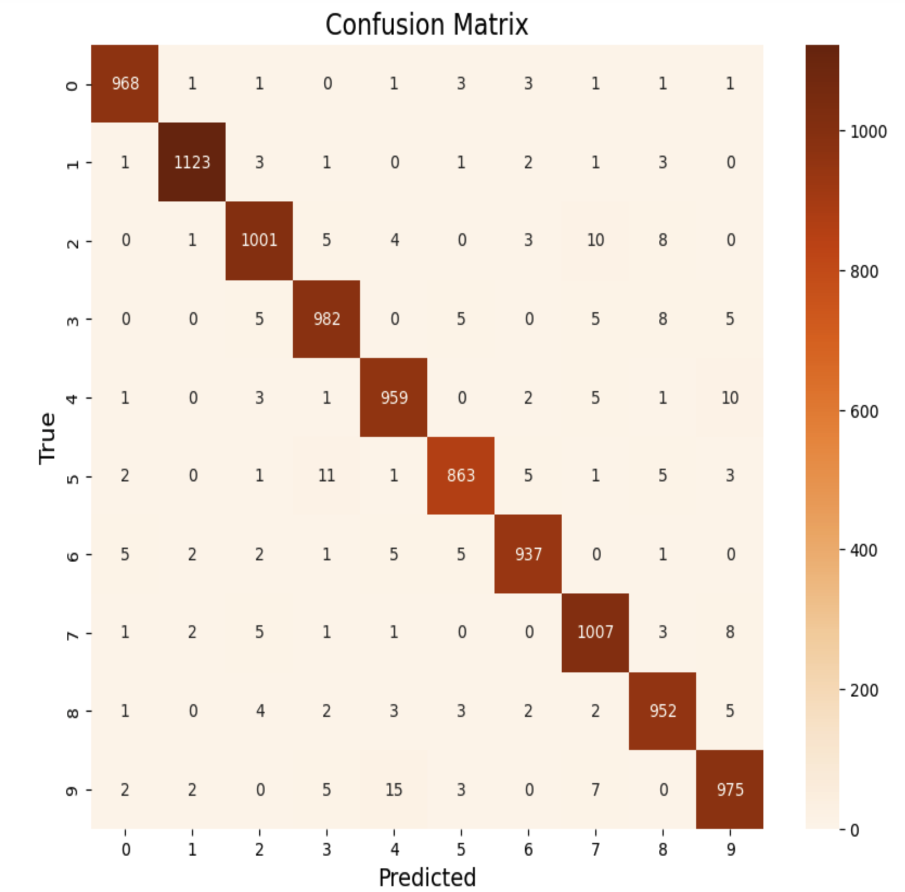


Figure 3.11: Train Accuracy and Validation Accuracy for each epoch

The confusion matrix is a matrix representation that summarizes the predictions made. The data shown illustrates the number of accurate and inaccurate predictions for each class. This helps in comprehending the classes the model misidentifies as belonging to a different class. The confusion matrix of the model,



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	968	1	1	0	1	3	3	1	1	1
1	1	1123	3	1	0	1	2	1	3	0
2	0	1	1001	5	4	0	3	10	8	0
3	0	0	5	982	0	5	0	5	8	5
4	1	0	3	1	959	0	2	5	1	10
5	2	0	1	11	1	863	5	1	5	3
6	5	2	2	1	5	5	937	0	1	0
7	1	2	5	1	1	0	0	1007	3	8
8	1	0	4	2	3	3	2	2	952	5
9	2	2	0	5	15	3	0	7	0	975

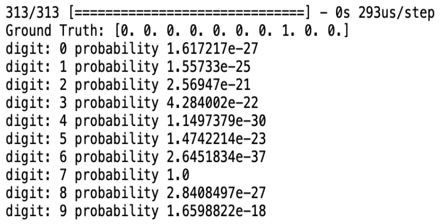
Figure 3.12: Confusion Matrix of Neutral Network Model

Since the accuracy of the model is very high, it will usually predict correctly. In the example below, the model predicts 7 when input 7 is used.

Python code 8:

```
predictions = model.predict(X_test)
index = 0 #input:7
print('Ground Truth:', y_test[index])
for i in range (10):
    print('digit:', i, 'probability', predictions[index][i])
```

Output:



```
313/313 [==============================] - 0s 293us/step
Ground Truth: [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
digit: 0 probability 1.617217e-27
digit: 1 probability 1.55733e-25
digit: 2 probability 2.56947e-21
digit: 3 probability 4.284002e-22
digit: 4 probability 1.1497379e-30
digit: 5 probability 1.4742214e-23
digit: 6 probability 2.6451834e-37
digit: 7 probability 1.0
digit: 8 probability 2.8408497e-27
digit: 9 probability 1.6598822e-18
```

Figure 3.13: Output of Python code 8

Consequently, a neural network model was implemented using the Python programming language.

3.7 Reversing Model

This is the main question "How to reach input if weight, bias and output are unknown?". When the activation function is ignored, it is quite simple to reverse.

$$output = (input * weight) + bias$$

Reverse it,

$$input = (output - bias)/weight$$

Although it is difficult to estimate the activation function, there are several research on attacks, and by knowing the weights and biases, it is possible to reach the input data. ReLU is not an invertible function, therefore things may become a little complicated when it's employed as the activation function. ReLU eliminates negative inputs while maintaining positive inputs. As a result, it could be difficult to determine which input value was obtained with the output value if it is positive. Reversal may be achieved in a few specific instances in a simple model using the ReLU layer. Anyone may estimate the proportionality of the result to the input, for instance, if the output value is positive and the weights used to determine the input are also positive.

When the computational information of the model is hidden, its resistance to these attacks will increase. The weight, bias, and inputs of the model are often hidden using the supported technique, which is seen to be the most accurate way to close these weaknesses in security.

CHAPTER 4

IMPLEMENTATION OF ZKML PROTOTYPE

This thesis presents a ZKML example for Neural network models based on the problems mentioned in "Importance and relevance of privacy in ML". Via Circom language, it proves that the neural network model belongs to the model created without giving its weight, bias, and inputs to the other party.

In the previous chapter, the input comes to the model, the weights are multiplied, and bias is added to it in the neural networks model. The result of this operation is the input of the next layer. In the neural network example given above, there were two hidden layers, and the *ReLU()* activation function was used between the hidden layers. Finally, the probabilities of the classes were determined with the *Softmax()* function. The model output was also reached with the *Argmax()* function.

The flowchart shown below shows the formation of final weights via iterative processes. The final model result is obtained when the indicated functions are applied to these weights.

This is the Neural Network Modelling process,

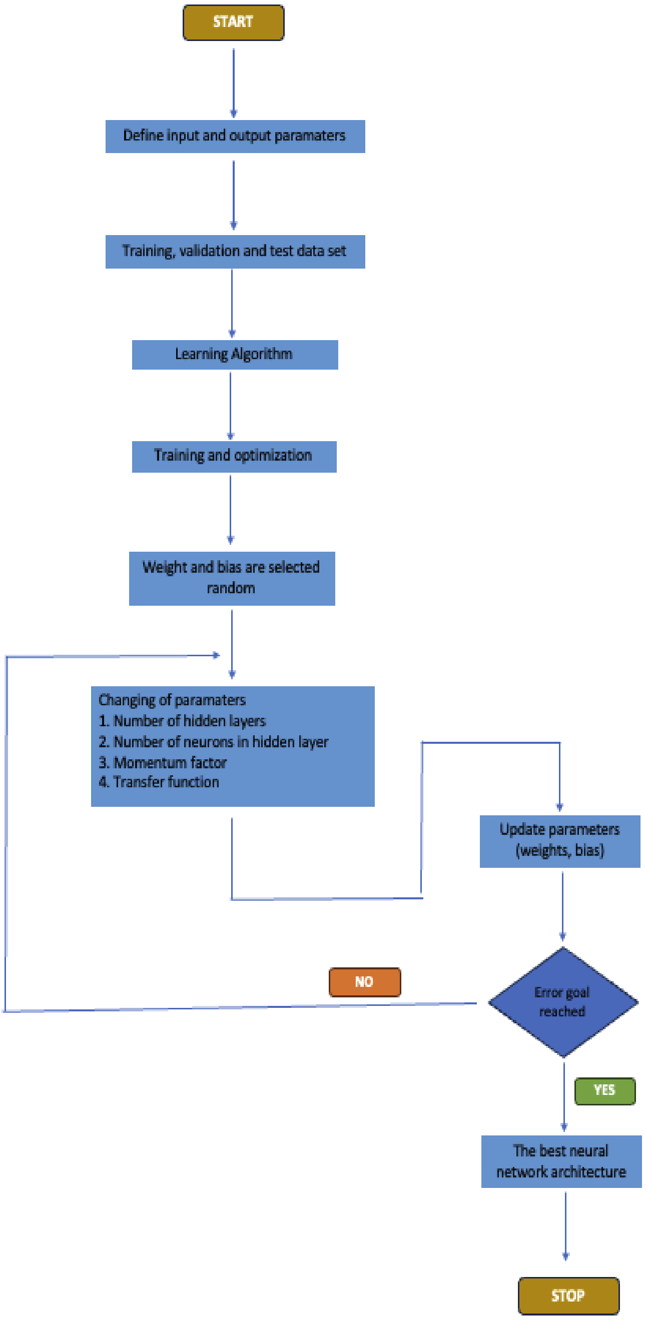

START
Define input and output paramaters
Training, validation and test data set
Learning Algorithm
Training and optimization
Weight and bias are selected random
Changing of paramaters
1. Number of hidden layers
2. Number of neurons in hidden layer
3. Momentum factor
4. Transfer function
Update parameters (weights, bias)
NO
Error goal reached
YES
The best neural network architecture
STOP

Figure 4.1: Neutral Network Flowchart

The weight, bias and input in the final version of the model should remain hidden. Model calculations can be done using CIRCOM without revealing this information by using Groth16. Terminal or zkREPL can be used as a compiler. Both options will be presented in this thesis.

4.1 CIRCOM and Snarkjs

Domain-specific languages (DSLs), such as Circom, aim to facilitate the development of ZKP applications by providing a witness generator and a constraint system based on the R1CS paradigm. The latter may be inputted into a zkSNARK generator in order to generate a prover and verifier. The use of the three aforementioned artifacts, namely the witness generator, prover, and verifier, may be employed in the construction of a ZK application. The CIRCOM compiler mostly employs the Rust programming language and is publicly available as an open-source software. The language is specifically designed to operate at a low-level, closely resembling the structure and functionality of circuits.

Snarkjs is a JavaScript version of zksnarks, and the library provides for the creation of proofs, verification part, and trusted setup.

4.2 Terminal

In this method, Using the Circom 2 Document [12], the script created for the model can be verify in the terminal step by step. Also, the following table shows which files are created using the terminal and what are used as inputs.

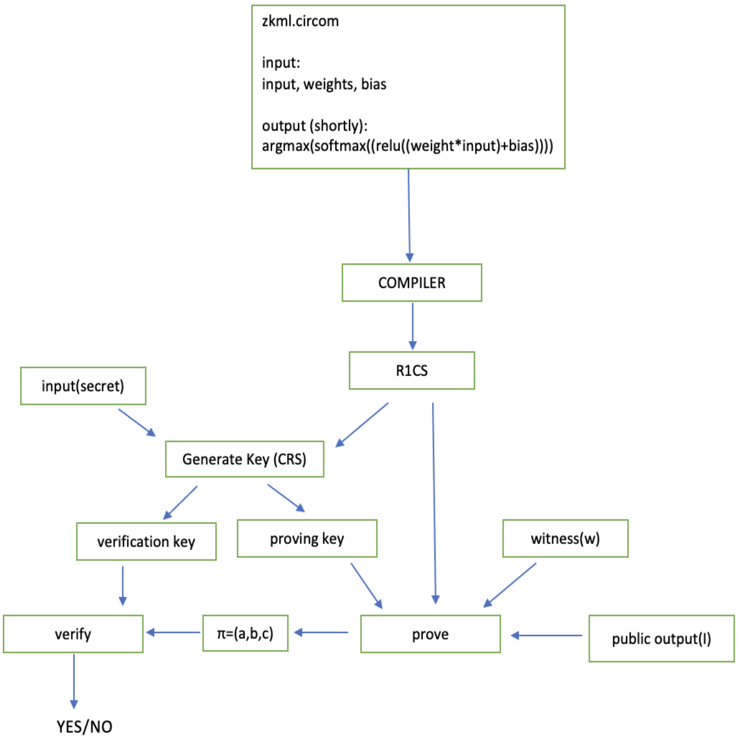


Figure 4.2: Circom and Snarkjs

4.2.1 Step 1: Installation

The primary instrument used is the Circom compiler, implemented in the Rust programming language. To make Rust accessible on your system, it is possible to install Rustup. For users operating on Linux or macOS systems, accessing the terminal and input the following command is recommended. With the following commands, a folder will be created in the name of Circom on the computer. The necessary files can be made in this folder, and the process can be completed.

Terminal code 1:

```
curl --proto '=https' --tlsv1.2 https://sh.rustup.rs -sSf | sh
git clone https://github.com/iden3/circom.git
cd circom
cargo build --release
cargo install --path circom
npm install -g snarkjs
```

4.2.2 Step 2: Writing Circuits

It is aimed to explain the logic by choosing the numbers that are small integers. There are five inputs, two hidden layers, each hidden layer contains two nodes and lastly, it can be thought of as having three classes. It includes *Relu()*, *Softmax()* and *Argmax()* functions with final weights. The circuit specially created for neural networks is as follows:

```
pragma circom 2.1.4;
function relu (x) {
  return x > 0 ? x : 0;
}
function max (x, y) {
  return x > y ? x : y;
}
function power (x, n) {
  var result = 1;
  for(var i = 0; i < n; i++){
    result = result * x;
}
  return result;
}
template weight(){
  signal input in[5];
  signal input w1[10];
  signal input w2[4];
  signal input w3[6];
  signal input bias1;
  signal input bias2;
  signal input bias3;
  signal hidden_1[2];
  signal hidden_2[2];
  signal out[3];
  signal temp1;
  signal temp2;
  signal temp3;
  signal temp4;
  signal temp5;
  signal temp6;
  signal temp7;
  signal temp8;
  signal temp9;
  signal temp10;
  signal softmax_sum;
  signal max_value;
  signal final1;
  signal final2;
  signal final3;
  signal output result1;
  signal output result2;
  signal output result3;
  signal d;
  d <-- 1;
  signal e;


e <-- (272/100);
// Hidden 1 result
temp1 <-- relu((in[0]*w1[0]) + (in[1]*w1[2]) + (in[2]*w1[4])
+ (in[3]*w1[6]) + (in[4]*w1[8]) + bias1);
hidden_1[0] <== temp1 * d;
temp2 <-- relu((in[0]*w1[1]) + (in[1]*w1[3]) + (in[2]*w1[5])
+ (in[3]*w1[7]) + (in[4]*w1[9]) + bias1);
hidden_1[1] <== temp2 * d;

// Hidden 2 result
temp3 <-- relu((temp1*w2[0]) + (temp2*w2[2]) + bias2);
hidden_2[0] <== temp3 * d;
temp4 <-- relu((temp1*w2[1]) + (temp2*w2[3]) + bias2);
hidden_2[1] <== temp4 * d;

// output
temp5 <-- relu((temp3*w3[0]) + (temp3*w3[2]) + bias3);
out[0] <== temp5 * d;
```

```
temp6 <-- relu((temp4*w3[1]) + (temp4*w3[3]) + bias3);
out[1] <== temp6 * d;
temp7 <-- relu((temp4*w3[2]) + (temp4*w3[5]) + bias3);
out[2] <== temp7 * d;

// softmax function
softmax_sum <--  temp5 + temp6 + temp7;
temp8 <-- power(e,temp5)/softmax_sum;
temp9 <-- power(e,temp6)/softmax_sum;
temp10 <-- power(e,temp7)/softmax_sum;

//Finally result
 max_value <-- max(max(temp8,temp9),temp10);
 if (max_value == temp8) {
 final1 <-- temp5;
 } else if (max_value == temp9){
 final2 <-- temp6;
 } else{


final3 <-- temp7;
   }
   result1 <== final1 * d;
   result2 <== final2 * d;
   result3 <== final3 * d;
}
component main = weight();
/* INPUT = {
    "in": [1, 2, 3, 4, 5],
    "w1": [5, 5, 5, 3, 1, 5, 1, 2, 1, 1],
    "w2": [1, 4, 3, 2],
    "w3": [1, 5, 3, 2, 3, 0],
    "bias1": "1",
    "bias2": "1",
    "bias3": "1"
} */
```

A template named "weight()" has been developed in the field of circuit design. In order to do this task, it is necessary to create a file with the code provided below. Once the arithmetic circuit has been constructed using the circom programming language, it is necessary to save the resulting circuit in a file with the .circom extension.

In this particular instance, a file named zkml.circom is generated. The current moment necessitates the compilation of the circuit in order to get a system of arithmetic equations that accurately represents it. As a consequence of the compilation process, then get programs that may be used to calculate the witness. The instruction is as follows:

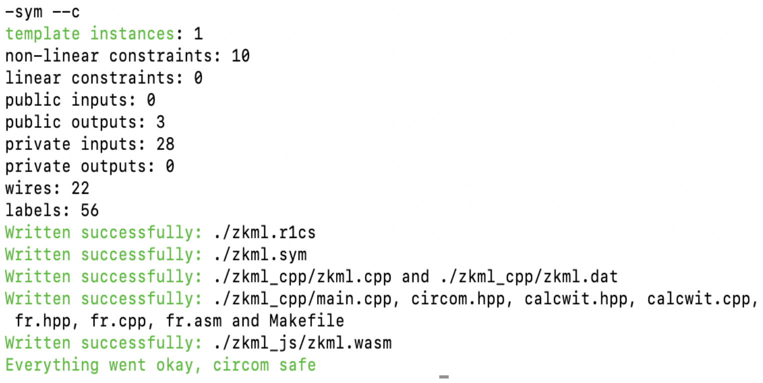
Terminal code 2:

```
circom zkml.circom --r1cs --wasm --sym --c
```

In this scenario, an R1CS file is generated using the "-r1cs" option. Additionally, a folder containing the files generate_witness.js, zkml.wasm, and witness_calculator.js is created using the "–wasm" option.

An image of the terminal that shows both the public and private input/output is also shown below. Additionally, it contains recently made files.

Output:



```
-sym --c
template instances: 1
non-linear constraints: 10
linear constraints: 0
public inputs: 0
public outputs: 3
private inputs: 28
private outputs: 0
wires: 22
labels: 56
Written successfully: ./zkml.r1cs
Written successfully: ./zkml.sym
Written successfully: ./zkml_cpp/zkml.cpp and ./zkml_cpp/zkml.dat
Written successfully: ./zkml_cpp/main.cpp, circom.hpp, calcwit.hpp, calcwit.cpp,
 fr.hpp, fr.cpp, fr.asm and Makefile
Written successfully: ./zkml_js/zkml.wasm
Everything went okay, circom safe
```

Figure 4.3: Image of Terminal Output 1

4.2.3 Step 3: Computing Witness

A file called input.json needs to be created, which contains inputs written in standard json format. All input parameters must have values in it as well.

```
{
    "in": [1, 2, 3, 4, 5],
    "w1": [5, 5, 5, 3, 1, 5, 1, 2, 1, 1],
    "w2": [1, 4, 3, 2],
    "w3": [1, 5, 3, 2, 3, 0],
    "bias1": "1",
    "bias2": "1",
    "bias3": "1"
}
```

Navigate to the directory named "zkml_js". Proceed to include the provided input into a file named "input.json". Finally, launch the program.

Before creating the proof, calculation of all the circuit's signals satisfy all of the circuit's constraints. *Wasm* module is produced by Circom, which helps in doing this task, for that.

The system just need to provide a file with the inputs, and the module will run the circuit and compute all of the intermediate signals and the output using the created Wasm binary and three JavaScript files. Witness is the collective term for a collection of inputs, intermediate signals, and outputs. The witness is now calculated, and a binary file witness.wtns is created with it in a snarkjs-acceptable format. The witness.wtns file will be created by the command. This file has been encoded in a binary format compatible with snarkjs, the program uses to produce the actual proofs.

Terminal code 3:

```
node generate_witness.js zkml.wasm input.json witness.wtns
```

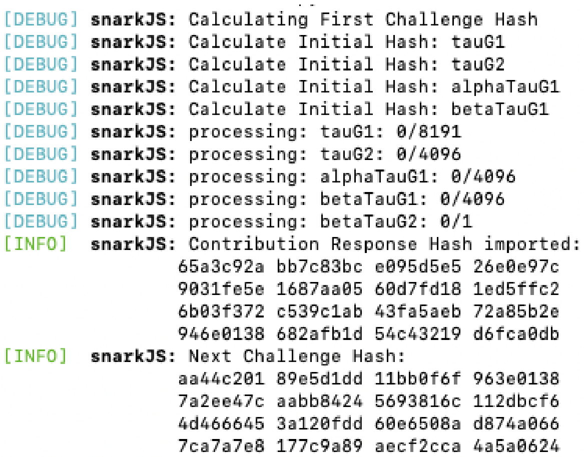
4.2.4 Step 4: Proving circuits (Groth16)

The first command activates the potency of the tau ritual. The *new* command is used to start a fresh phase, with the first argument after *new* denoting the curve type, hence specifying the Groth-16 parameter. In this context, we specify the selection of the curve, namely the $bns128$ curve. The subsequent parameter, denoted as 'a' and taking the value of 12 in this particular scenario, establishes an upper limit of 2^a on the number of constraints that the configuration has the capacity of accepting. The pot12_0000.ptau output file has been successfully configured. The second command serves as a significant addition to the ceremonial proceedings. A new file named pot12_0001.ptau will be generated, which will serve as a precise copy of the original file. Additional text is included to improve the level of entropy.

Terminal code 4:

```
snarkjs powersoftau new bn128 12 pot12_0000.ptau -v
snarkjs powersoftau contribute pot12_0000.ptau pot12_0001.ptau
--name="First contribution" -v
```

Output:



```
[DEBUG] snarkJS: Calculating First Challenge Hash
[DEBUG] snarkJS: Calculate Initial Hash: tauG1
[DEBUG] snarkJS: Calculate Initial Hash: tauG2
[DEBUG] snarkJS: Calculate Initial Hash: alphaTauG1
[DEBUG] snarkJS: Calculate Initial Hash: betaTauG1
[DEBUG] snarkJS: processing: tauG1: 0/8191
[DEBUG] snarkJS: processing: tauG2: 0/4096
[DEBUG] snarkJS: processing: alphaTauG1: 0/4096
[DEBUG] snarkJS: processing: betaTauG1: 0/4096
[DEBUG] snarkJS: processing: betaTauG2: 0/1
[INFO]  snarkJS: Contribution Response Hash imported:
                65a3c92a bb7c83bc e095d5e5 26e0e97c
                9031fe5e 1687aa05 60d7fd18 1ed5ffc2
                6b03f372 c539c1ab 43fa5aeb 72a85b2e
                946e0138 682afb1d 54c43219 d6fca0db
[INFO]  snarkJS: Next Challenge Hash:
                aa44c201 89e5d1dd 11bb0f6f 963e0138
                7a2ee47c aabb8424 5693816c 112dbcf6
                4d466645 3a120fdd 60e6508a d874a066
                7ca7a7e8 177c9a89 aecf2cca 4a5a0624
```

Figure 4.4: Image of Terminal Output 2

The contributions to the powers of tau are included in the file "pot12_0001.ptau". We can now continue with Phase 2.

Terminal code 5:

```
snarkjs powersoftau prepare phase2 pot12_0001.ptau pot12_final.ptau -v
```

The beginning of the second phase involves the calculation of a Groth16 CRS. The outcome of this stage is contingent upon the R1CS associated with the issue, and the resultant calculation is stored in the file named zkml_0000.zkey.

Terminal code 6:

```
snarkjs groth16 setup zkml.r1cs pot12_final.ptau zkml_0000.zkey
snarkjs zkey contribute zkml_0000.zkey zkml_0001.zkey
--name="1st Contributor Name" -v
snarkjs zkey export verificationkey zkml_0001.zkey verification_key.json
```

Generating a Proof, two files are created and their name is "proof.json" and "public.json". The "proof.json" file will include the representation of the real proof via three curve points like π, while the "public.json" will keep the values of the instance.

Terminal code 7:

```
snarkjs groth16 prove zkml_0001.zkey witness.wtns proof.json public.json
```

"proof.json" and "public.json" includes them,

proof.json:

```
{
 "pi_a": [
  "110007107169879889684540407167293033402970061
  14283318268370750364664387376660",

  "309320261999654173676816627378383082772374584
  4104446358026888656502754350723",
  "1"
 ],
 "pi_b": [
  [
   "11918365174906303943622045930621900258309023
   823041378998581306592065688446777",
   "128095475342160232544087512511108698705645 90
   1879397005133280046339688039230 92"
  ],
  [
   "8004609285971355955755079701122137860216 5717
   2068360995566706873806317766 6908",

   "2709771322468846318857905715836485636916 0759
   827397823740848267594424463 39020"
  ],
  [
   "1",
   "0"
  ]
 ],
 "pi_c": [
 "79379096632220649292840241801650882688301 62025
 43664596609601914527677312 5208",

  "557554480120459382490238839489416483112 135168
  1983723104402994446372992667414",
  "1"
 ],
 "protocol": "groth16",
 "curve": "bn128"
}
```

public.json:

```
[
 "0",
 "0",
 "580"
]
```

According to previous chapter, it is possible to get a verifier key from the CRS. In order to validate the evidence proof.json against the instance public.json, the verifier uses the verification key verification_key.json and applies the verification algorithm provided by Snark.js. For verification of the proof, please perform the provided command:

Terminal code 8:

```
snarkjs groth16 verify verification_key.json public.json proof.json
```

This is the verification key:

```
{
 "protocol": "groth16",
 "curve": "bn128",
 "nPublic": 3,
 "vk_alpha_1": [
  "4729492488277101573624767949545658868784797
  357062990795349388016965269418084",
  "3519788845822786375959133130017828954038670
  591329401705054971621899448324453",
  "1"
 ],
 "vk_beta_2": [
  [
   "2211567741354393684859489156796815846003960
   198091728054854525744378230062764",
   "2117347711612772369432770650726177129468227
   6118108574822440157077141653111699"
  ],
  [
   "1873313006285346621024981202713339211931864
   5238473227048294205171823217362069",
   "4797524190777451848383007268074776485718403
   527558482577743075769444636566503"
  ],
  [
   "1",
  "0"
 ]
],
"vk_gamma_2": [
 [
  "1085704699902305713594457076223282948137075635957851808699051999328565585 2781",
  "1155973203298638710799100402139228578392581286182119253091740315145239180 5634"
 ],
 [
  "8495653923123431417604973247489272438418190587263600148770280649306958101930",
  "4082367875863433681332203403145435568316851327593401208105741076214120093531"
 ],
 [
  "1",
  "0"
 ]
],
"vk_delta_2": [
 [
  "9724809083748902525604245066648348625457035767082879279067774638204810690324",
  "2646889759062801914326140639259238994139888950247818315918806909778144895798"
 ],
 [
  "4566851435307016323262529567210338234304525066048603040022933477307826545101",
  "8306598626329780972040057388429883486476915647196706190498690268313590032485"
 ],
 [
  "1",
  "0"
  ]
 ],
 "vk_alphabeta_12": [
  [
   [
    "104642615067670691924252433624603397552901673
    18153616356982569750227841548246",
    "121229859293745860406627637035876638650783690
    9567194958284859138954201640209 8"
   ],
   [
    "869231136248303407211384718221998603185332501
    98620248498267554490304361501 36",
    "201143524442190322061178062613881205096263846
    804045992077072420246334897408 47"
   ],
   [
    "355138887918081564648749613811070183298341688
    06087803780678674087794341121 42",
    "202826376573411813627760472527532748371157685
    733726564301103572609442089946 48"
   ]
  ],
  [
   [
    "1490678063775391402629240039023942345063077265
    1030575768641188076541690898668",
    "1962259826236688758527866632079693813413913495
    345422737857484732262906095268 0"
   ],
   [
    "1369198275254817322287894969646425249660901653
    7224773158256779460421550231222",
    "1860430347125827469874489448683432310467211289
    222068356455235618949344287612 6"
   ],
   [
   "2053437585644425978664327165107380557777803553
   3315274687155162429429782147853",
   "7063303924819833907610767409645562594420267780
   421376151950063177753567889420"
  ]
 ]
],
"IC": [
 [
  "1829467938080895318130495163688871163165867197020
  2584968773859462936054541948",
  "2608364181853022527170046095816881629294269692771
  69341008081399982810302841",
  "1"
 ],
 [
  "9719129603349410940549348837776995197841453378448
  64198688314446727599526169",
  "4512776640736705467208617224041688589287262121895
  622939282424254541231469850",
  "1"
 ],
 [
  "2060368352706189074621306280916887496521112096241
  3162366074125393659932303533",
  "1385525187970767475703340055916997047934926200744
  6265478798849432209294152102",
  "1"
 ],
 [
  "3423568362570347463859662006736774881910073822810
  615890955691151161273649666",
  "8547419220301671118511551893955286103775608631385
  046904056357637600124249523",
  "1"
 ]
]
```

```
}
```

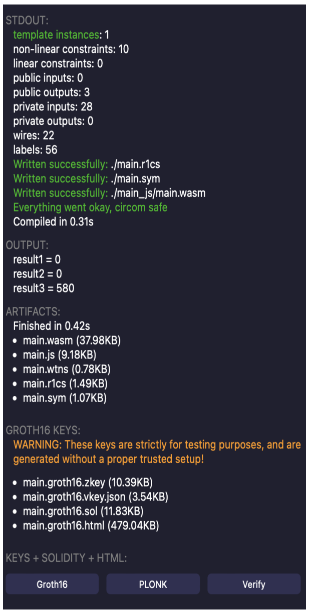
At the end of the verification, the terminal says "Ok".

4.3 ZkREPL

Formal languages may be viewed and expressed as claims of membership or knowledge, in which algebraic circuits and R1CS perform crucial roles in defining and characterizing these languages. To enable the creation of intricate statements, it is essential to possess a compiler framework capable of converting high-level programming languages into arithmetic circuits and their corresponding R1CS.

These programming languages facilitate the development and evaluation of arithmetic circuits more naturally and efficiently, relieving developers of the complexities associated with R1CS representations. Consequently, individuals can focus on the logical aspects of the courses they want to construct since the compiler assumes responsibility for the remaining tasks.

ZkREPL is a compiler that aims to combine ZKP systems with more complex structures based on the Circom language. By clicking, files can be downloaded and saved. You can check if the circuit is working here in much less time.



```
STDOUT:
  template instances: 1
  non-linear constraints: 10
  linear constraints: 0
  public inputs: 0
  public outputs: 3
  private inputs: 28
  private outputs: 0
  wires: 22
  labels: 56
  Written successfully: ./main.r1cs
  Written successfully: ./main.sym
  Written successfully: ./main_js/main.wasm
  Everything went okay, circom safe
  Compiled in 0.31s
OUTPUT:
  result1 = 0
  result2 = 0
  result3 = 580
ARTIFACTS:
  Finished in 0.42s
  • main.wasm (37.98KB)
  • main.js (9.18KB)
  • main.wtns (0.78KB)
  • main.r1cs (1.49KB)
  • main.sym (1.07KB)

GROTH16 KEYS:
  WARNING: These keys are strictly for testing purposes, and are
  generated without a proper trusted setup!

  • main.groth16.zkey (10.39KB)
  • main.groth16.vkey.json (3.54KB)
  • main.groth16.sol (11.83KB)
  • main.groth16.html (479.04KB)

KEYS + SOLIDITY + HTML:
```

Figure 4.5: The output of zkREPL

Proof:

```
{"pi_a":["94350128754355673222095400400330664411496
125061450982240510754020396759191O1",
"74157313413130936790677802135537591262076993532024
91693102669091419299301911","1"],
"pi_b":[["47283443669100825473176759496621888200518896O4003
28216995191760332448294258l",
"18870578681852114049847427623127733941542244419344 9
86786400115243637129821253"],
["68633548667596838495798105394352550571638991294887
37834158283220352039515125",
"58073428839915456281182599851532715650667969151O282
20538257069540128921913O5"],["1","0"]],
"pi_c":["75213201298452936811147594002147850944841832360051
61493719828393082490808603",
"17532615846116768740871697818121082795452231772862 8
59716079769737357186508756","1"],"protocol":"groth16","curve":"bn128"}
```

Public Output:

```
["0","0","580"]
```

As mentioned before, the variable with the largest ratio in this context is the projected result.

CHAPTER 5

CONCLUSION

The neural network model has been proven by using each of the options suggested that which are ZkREPL or terminal, and not sharing the input, weight, and bias. The results show that our tried-and-true method of confirming neural network models using the Groth16 restart protocol while keeping the details of the base parts secret works. This suggested method is a new idea that handles security and privacy concerns well, which could make neural network-based models more reliable.

The study's results show that neural network models can be made with different internal settings when both data security and model security must be kept. This method hides parts of the model so that they can't be changed or abused from the outside. This method can also make the model safer to use in the place where it is made, and it can widen the range of security measures.

The findings acquired have the potential to enhance the security of systems based on neural networks, while also carrying significant consequences in the domains of machine learning and cryptography. Further investigation should be conducted to thoroughly examine the potential of this method on bigger datasets and other machine learning architectures in the future.

In summary, this research provides a novel viewpoint about the security aspects of systems based on neural networks. There is an expectation that future research efforts in this domain will be focused on advancing the methodology and expanding its range of applications.

5.1 Limitations

- The rationale for the need of multiplying any circuit result by 1 remains unclear. This observation suggests a potential deficiency in the user's understanding or methodology to substantiate their claim, notwithstanding the circuit's theoretical validity.
- Due to its limited applicability to quadratic equations, the circuit precluded the consideration of a broader range of equations. The restricted range of applicability of the circuit constrained its use in addressing diverse mathematical issues, hence prompting the exploration of alternate methodologies for dealing with more intricate equations.
- Because of the inherent limitation of the Circom language in accommodating decimal values, other approaches were required to effectively handle such numerical representations. Consequently, indirect methodologies and intricate computations were used throughout the procedure. This problem is fixed with the following function,

```
function power (x, n) {
  var result = 1;
  for(var i = 0; i < n; i++){
    result = result * x;
}
  return result;
```

5.2 Future Work

It is important to assess the wider potential of the technique proposed in this thesis by subjecting it to testing using an array of neural network models. This will provide a comprehensive evaluation of the method and facilitate a deeper understanding of its inherent limits. By conducting experiments with various model types, depths, and topologies, researchers may strive to ascertain the optimal conditions in which the approach exhibits superior performance, as well as identify situations where additional enhancements are necessary. These studies provide a more comprehensive perspective on the practical uses of the approach in real-world neural network contexts.

Furthermore, within the context of extended validation, it is possible to enhance the procedure in order to verify not only the weight, bias, and inputs of the neural network model but also the intrinsic activations at various levels. For example, the number of layers, the number of nodes, and the activation functions used can also be hidden.

Further investigation and refinement of the approach presented in this thesis with regard to its consequences on data privacy and security may be considered as a viable avenue for future research. In situations when the approach is deemed suitable, it is crucial to evaluate any vulnerabilities that may expose the underlying architecture of the model or result in breaches of privacy. In this particular context, the objective is to simulate several attack routes and evaluate the effectiveness of defensive measures against these assaults.

The methodology used in this thesis is not necessarily restricted just to the validation of neural network models. Further investigation might be conducted to examine the potential for applying a comparable methodology to other models, such alternative machine learning techniques or statistical models. It is crucial to showcase the method's potential for broader use as a comprehensive validation methodology.

REFERENCES

[1] Arnsx, A. (2019) First neural network for beginners explained (with code). url: https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf [Accessed on 15 Aug, 2023].

[2] Ayodele, T. O. (2010). Types of machine learning algorithms. New advances in machine learning, 3, 19-48.

[3] Bellés-Muñoz, M., Isabel, M., Muñoz-Tapia, J. L., Rubio, A., & Baylina, J. (2022). Circom: A Circuit Description Language for Building Zero-knowledge Applications. IEEE Transactions on Dependable and Secure Computing.

[4] Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., & Ward, N. P. (2019). Aurora: Transparent succinct arguments for R1CS. In Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38 (pp. 103-128). Springer International Publishing.

[5] Benarroch, D., Brandão, L., Maller, M., and Tromer, E. Pub. (2022). ZKProof Community Reference. Version 0.3. Ed. by by zkproof.org. Updated versions at https://docs.zkproof.org/reference [Accessed on 15 Jan, 2023]

[6] Chen, T., Lu, H., Kunpittaya, T., & Luo, A. (2022). A review of zk-snarks. arXiv preprint arXiv:2202.06877.

[7] Ghodsi, Z., Gu, T., & Garg, S. (2017). Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. Advances in Neural Information Processing Systems, 30.

[8] Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., & Wernsing, J. (2016). Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In International conference on machine learning (pp.

201-210). PMLR.

[9] Groth, J. (2005). Non-interactive zero-knowledge arguments for voting. In Applied Cryptography and Network Security: Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005. Proceedings 3 (pp. 467-482). Springer Berlin Heidelberg.

[10] Groth, J. (2016). On the size of pairing-based non-interactive arguments. In Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35 (pp. 305-326). Springer Berlin Heidelberg.

[11] Gudikandula, P. (2019). A Beginner Intro to Neural Networks. url: https://medium.com/@purnasaigudikandula/a-beginner-intro-to-neural-networks-543267bda3c8 [Accessed on 17 Aug, 2023].

[12] Iden3, “CIRCOM: Circuit compiler for zero-knowledge proofs”, GitHub, 2020. [Online]. Available: https://github.com/iden3/circom [Accessed on 15 Aug, 2023].

[13] Kang D., & Gan E. Bridging the Gap: How ZK-SNARKs Bring Transparency to Private ML Models with zkml. 2023. url: https://medium.com/@danieldkang/bridging-the-gap-how-zk-snarks-bring-transparency-to-private-ml-models-with-zkml-e0e59708c2fc [Accessed on 4 Aug, 2023].

[14] Lee, S., Ko, H., Kim, J., & Oh, H. (2020). vcnn: Verifiable convolutional neural network based on zk-snarks. Cryptology ePrint Archive.

[15] libsnark, https://github.com/scipr- lab/libsnark [Accessed on 1 Aug, 2023].

[16] Ma, C., Wojtowytsch, S., & Wu, L. (2020). Towards a mathematical understanding of neural network-based machine learning: what we know and what we don’t. arXiv preprint arXiv:2009.10713.

[17] Mahesh, B. (2020). Machine learning algorithms-a review. International Journal of Science and Research (IJSR).[Internet], 9(1), 381-386.

[18] MNIST, http://yann.lecun.com/exdb/mnist/ [Accessed on 5 Aug, 2023].

[19] Muñoz-Tapia, Jose L.; Belles, Marta; Isabel, Miguel; Rubio, Albert; Baylina, Jordi (2022). CIRCOM: A Robust and Scalable Language for Building Complex Zero-Knowledge Circuits. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.19374986.v1 [Accessed on 17 Aug, 2023].

[20] Nitulescu, A. (2020). zk-SNARKs: a gentle introduction.

[21] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors." in Proc. of USENIX Security'16, 2016, pp. 619– 636.

[22] Paar, C. 1963-. (2009). Understanding cryptography : a textbook for students and practitioners. Berlin ; London :Springer.

[23] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in Proc. of ICML'16, 2016, pp. 201–210.

[24] Raffo, D. (2002). Digital Certificates and the Feige-Fiat-Shamir zero-knowledge protocol. Master of Science in Computer Science Thesis, Université de Marne la Vallée, France.

[25] Ray, S. (2019). A quick review of machine learning algorithms. In 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon) (pp. 35-39). IEEE.

[26] Ray, S. (2019). A Quick Review of Machine Learning Algorithms. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). https://doi.org/10.1109/comitcon.2019.8862451 [Accessed on 17 Aug, 2023].

[27] Richter, M. (2023). The MoonMath Manual to ZK-SNARKS v1.1.1. Least Authority Privacy Matters

[28] Saxena, A. (2020). Building a simple neural network from scratch. Medium. https://towardsdatascience.com/building-a-simple-neural-network-from-scratch-a5c6b2eb0c34 [Accessed on 17 Aug, 2023].

[29] Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. Towards Data Sci, 6(12), 310-316.

[30] Shinde, P. P., & Shah, S. (2018). A review of machine learning and deep learning applications. In 2018 Fourth international conference on computing communication control and automation (ICCUBEA) (pp. 1-6). IEEE.

[31] Smart, N. P. (2016). Cryptography Made Simple. (Information Security and Cryptography). Springer, https://doi.org/10.1007/978-3-319-21936-3 [Accessed on 21 Dec, 2022].

[32] Tuntas, R., & Dikici, B. (2016). An investigation on the aging responses and corrosion behaviour of A356/SiC composites by neural network: The effect of cold working ratio. Journal of Composite Materials, 50(17), 2323-2335.

[33] Venir, E., & Ruzza, L. (2022). Decentralized federated machine learning with blockchain and zero knowledge proofs.

[34] Vitalik Buterin. Quadratic Arithmetic Programs: from Zero to Hero. 2016. url: https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649 [Accessed on 15 Aug, 2023].

[35] Yagnadeepxo. (2023, April 26). The beginner's guide to ZK-snark: Setting up your first proof system. DEV Community. https://dev.to/yagnadeepxo/the-beginners-guide-to-zk-snark-setting-up-your-first-proof-system-3le3 [Accessed on 15 Aug, 2023].

[36] Yu, W., Xu, M., Yu, D., Cheng, X., Hu, Q., & Xiong, Z. (2022, November). zk-PCN: A Privacy-Preserving Payment Channel Network Using zk-SNARKs. In 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC) (pp. 57-64). IEEE.

[37] Zapechnikov, S. (2020). Privacy-preserving machine learning as a tool for secure personalized information services. Procedia Computer Science, 169, 393-399.

[38] Zhang, J., Fang, Z., Zhang, Y., & Song, D. (2020). Zero knowledge proofs for decision tree predictions and accuracy. Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. https://doi.org/10.1145/3372297.3417278 [Accessed on 15 Aug, 2023].

[39] Zhao, L., Wang, Q., Wang, C., Li, Q., Shen, C., & Feng, B. (2021). Veriml: Enabling integrity assurances and fair payments for machine learning as a service. IEEE Transactions on Parallel and Distributed Systems, 32(10), 2524-2540.