

Malware Detection in Blockchain using CNN

by

Afreen Alam

17301038

Humaira Islam

17101045

Sadman Arif Wamim

17101041

Md. Tanjim Ahmed

17301146

Hasnat Siddiqi

17301186

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
BRAC University
January 2021

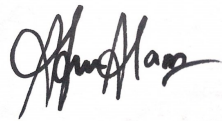
© 2021. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Afreen Alam
17301038




Humaira Islam
17101045



Sadman Arif Wamim
17101041



Md. Tanjim Ahmed
17301146



Hasnat Siddiqi
17301186

Approval

The thesis/project titled “Malware Detection in Blockchain using CNN” submitted by

1. Afreen Alam (17301038)
2. Humaira Islam (17101045)
3. Sadman Arif Wamim (17101041)
4. Md. Tanjim Ahmed (17301146)
5. Hasnat Siddiqi (17301186)

Of Fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 15, 2021.

Examining Committee:

Supervisor:
(Member)



Moin Mostakim
Lecturer
CSE Department
BRAC University

Program Coordinator:
(Member)



Md. Golam Rabiul Alam, PhD
Associate Professor
CSE Department
BRAC University

Head of Department:
(Chair)

Mahbub Alam Majumdar, PhD
Professor and Chairperson
Department of Computer Science and Engineering
BRAC University

Ethics Statement

We, Afreen Alam, Humaira Islam, Sadman Arif Wamim, Md. Tanjim Ahmed and Hasnat Siddiqi, hereby attest that for this thesis, Malware Detection in Blockchain using CNN, we have abided by the following rules-

1. This paper is an original work of all the authors, which has not been previously published elsewhere and is not being considered for publication elsewhere.
2. The paper reflects the authors' own research and analysis in a truthful and complete manner.
3. The paper properly credits the significant contributions of co-authors and co-researchers.
4. The results are placed in an appropriate manner in the context of prior and existing research.
5. The sources used in this paper are properly disclosed (correct citation).
6. We take full responsibility for all the work done as authors for this paper. The violation of the Ethical Statement rules may result in severe consequences.

We agree with the above statements and declare that this submission follows the policies of BRAC UNIVERSITY as outlined in the Guide for Authors and in the Ethical Statement.

Corresponding Authors' Full Name & Signature:



Afreen Alam
17301038



Humaira Islam
17101045



Sadman Arif Wamim
17101041



Md. Tanjim Ahmed
17301146



Hasnat Siddiqi
17301186

Abstract

The inherent decentralized nature and peer-to-peer system of the blockchain's popularity has been on the rise in recent times and is being adopted in various innovative applications. This technology claims to be one of the most secure inventions due to the employment of hash functions, which makes the data stored immutable. However, security issues concerning blockchains have been highlighted in recent reports, which begs the question: is the blockchain technology as invulnerable as it once claimed to be? These reports talk about malware injections which lead to data corruption, data theft as well as third parties gaining networking power. This has become a significant worry for security in the dynamic online world. To counter such security concerns, we propose a model which combines a convolutional neural network with a blockchain in order to prevent malicious data transactions and thus malware injection within a blockchain network. This convolutional neural network detects any malware that might be present in the data before a new block is created to be a part of the blockchain. We have compared two different CNN models: the VGG-16 architecture and a customized model with fewer layers. When integrated with our blockchain model, the VGG-16 convolutional neural network architecture achieves an accuracy of 90.3% while the custom model achieves an accuracy of 88.90%.

Keywords: Malware detection; Blockchain; Convolutional Neural Network

Dedication

We dedicate this thesis to our parents, who have supported us and sacrificed so much to get us this far in our lives.

We would also like to dedicate this thesis to our friends, who have cheered us on when things got difficult and have been an immense support throughout it all. This work is also dedicated to our supervisor who inspired us to think outside the box and put in the hard work effort to produce better results. We would also like to dedicate this work to BRAC University which has given us this amazing platform to learn and grow.

Acknowledgement

Firstly, all praise to Allah for whom we could successfully complete this thesis without any major interruption. We are all grateful to Him for keeping all of us healthy in these trying times.

Next, we are grateful to our supervisor Mr. Moin Mostakim sir as well as our co-supervisor Dr. Mohammad Iqbal Hossain sir, who have provided us with their invaluable time, support and help whenever we needed it. Our relentless effort to achieve our goal was always appreciated by our supervisor and co-supervisor who ensured that we never gave up.

We are thankful to our parents, without whom we could not have made it.

And lastly, we want to thank our beloved institution BRAC University as well as all the respectable faculties and staff members who have given us an insightful educational journey .

There were times when the tide was against us however through perseverance we have concluded our work after a long working year and hopefully made a significant impact.

Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iii
Abstract	iv
Dedication	v
Acknowledgment	vi
Table of Contents	vii
List of Figures	ix
List of Tables	x
Nomenclature	xi
1 Introduction	1
1.1 Introduction	1
1.2 Problem statement	2
1.3 Aim of study	2
1.4 Research methodology	2
1.5 Thesis outline	3
2 Related Work	4
2.1 Convolutional Neural Network	4
2.1.1 What is a convolutional neural network?	4
2.1.2 Work related to convolutional neural networks	4
2.2 Blockchain	6
2.2.1 What is a blockchain?	6
2.2.2 Work related to blockchain	7
2.3 Blockchain with Neural Networks	9
2.3.1 A combined approach	9
2.3.2 Work that combines both the technologies	9

3	Data collection and preprocessing	11
3.1	Dataset for training and testing the CNN model	11
3.2	Dataset for testing the model	12
3.2.1	Data collection	12
4	Proposed Model	13
4.1	CNN Layers and Functions	13
4.1.1	Convolution layer	13
4.1.2	Pooling layer	13
4.1.3	Dropout layer	14
4.1.4	Flatten layer	14
4.1.5	Fully connected layer	14
4.1.6	ReLU activation function	14
4.1.7	Sigmoid activation function	14
4.1.8	Cross entropy loss function	14
4.2	The custom CNN model	15
4.3	VGG-16	15
4.4	Blockchain	16
4.4.1	Genesis Block	17
4.4.2	Data Block	18
4.5	Combined model	20
5	Experimental Setup and implementation	22
5.1	Machine specifications for implementation	22
5.2	CNN	22
5.3	Flask API	22
5.4	File Conversion	23
5.5	Blockchain	24
5.6	Combined model	25
6	Result analysis	26
6.1	Results	26
6.2	Blockchain Output	29
6.3	Limitations	29
7	Conclusion and future work	30
7.1	Conclusion	30
7.2	Future work	31
	Bibliography	35

List of Figures

3.1	Quantity of Malware Images in Different Families	11
4.1	The architecture of the custom CNN model	15
4.2	The architecture of the VGG-16 model	16
4.3	Basic architecture of a blockchain	17
4.4	Topology of Private and Public blockchain	17
4.5	Architecture of a genesis block	18
4.6	Architecture of a data block	18
4.7	How the previous hash and current hash works	19
4.8	Workflow diagram of the model	21
5.1	Flask implementation	23
5.2	File conversion	23
5.3	Any file conversion to grayscale image	23
5.4	User interface	25
6.1	Custom CNN Model Results	26
6.2	VGG-16 CNN Model Results	26
6.3	Accuracy and loss comparison between the two CNN models	27
6.4	Percentage accuracy of custom model in classifying malware files from different malware families	28
6.5	Percentage accuracy of VGG-16 in classifying malware files from different malware families	28
6.6	Percentage accuracy in classifying malware files from non-malware files vs. Model	28
6.7	Detection of Malware	29
6.8	No malware detected, Blockchain is triggered	29

List of Tables

6.1	Time required to get server response	27
-----	--	----

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

σ Sigma

API Application Programming Interface

CNN Convolutional Neural Network

CORS Cross Origin Resource Sharing

DDoS Distributed Denial of Service

HTML Hypertext Markup Language

HTTP Hyper Text Transfer Protocol

JSON Javascript Object Notation

PHP Hyper Text Preprocessor

ReLU Rectified Linear Unit

SHA256 Secure Hash Algorithm

SQL Structured Query Language

VGG – 16 Visual Geometry Group

Chapter 1

Introduction

1.1 Introduction

In an ever increasing and evolving technology driven world, the increase of internet access has led to an aggressive increase in cybercrimes. Cybercrimes are generally achieved through malware attacks [1]. Malware which is short for malicious software, is defined as any software or program that is devised with the intention to accomplish diverse security attacks. Various malwares specialize in specific kinds of attacks. For example, viruses cause malfunction to the files or the system itself, trojans enable unauthorized access to the system, loss or theft of information, modification of data, spywares can monitor a user's activities without their knowledge, harvest data including user credentials etc. Such attacks can have severe long term effects on individual devices such as causing them to slow down or slow down a connection, keeping processes active which keeps devices from ever shutting down as well as system corruption. Large scale effects include harming business organizations by stealing or holding to ransom valuable business data. According to Cybersecurity Ventures [2], it is estimated that the damage caused by cybercrimes will exceed \$6 trillion USD by the year 2021. While according to SafeAtLast [3], corporations lose approximately 133,000 dollars due to ransomware assaults. It is now more crucial than ever to provide a safe and secure environment on the internet for protecting the privacy of people. After the release of the Bitcoin white paper by Satoshi Nakamoto in 2008, blockchain has been one of the most widely discussed ways to encrypt data storage and transmission in the form of decentralized, trustless, peer-to-peer systems [4]. Blockchain technology's unique properties make it appealing for usage in various fields like cryptocurrencies, smart contracts, communication systems, health care, Internet of Things, financial systems, censorship resistance, electronic voting distributed provenance and many more. The append-only nature of the blockchain means that transactions cannot be modified, while the transparency aspect allows the storage of publicly verifiable and undeniable records. The peer-to-peer system delivers ledger maintenance without the need for a centralized authority which diminishes the concern for single point-of-failure [5].

1.2 Problem statement

Blockchains, which were thought to be immune to cyberattacks are now being subjected to various malicious attacks. Some instances of attacks on blockchain are when in January 2019, an attacker had gained more than half of Ethereum Classic's blockchain's networking power and in September 2019, when the Glupteba malware hijacked Bitcoin's blockchain. Using the Electrum bitcoin wallet, the Glupteba malware sends bitcoin transactions that the attackers use to gain access to systems. It can also mine the privacy-specialized monero cryptocurrency and compromise the security of the accounts of Instagram users [6]. Bitfinix was the target of a distributed denial-of-service (DDoS) attack in June 2017 which led to its temporary suspension. Bitcoin and Ethereum were also subjected to such attacks which hindered the platforms' availability. Transaction stalls created by flooding the memory pools of Bitcoin resulted in a delay of 700 million USD worth bitcoins. The publicly verifiable property of blockchain makes it susceptible to dishonest activities. Two such attackers took advantage of this by stealing 460 million USD worth of bitcoins from a Japanese currency exchange called Mt.Gox [5]. A 51% attack where the attacker takes control of a majority of the hashrate on a blockchain, was launched against five blockchain-based cryptocurrencies leading to a loss of 5 million USD. The attackers were able to take over the entire blockchain network and perform double-spending where attackers can spend a digital currency more than once [5], [7]. Furthermore, utilizing the distributed nature of blockchains, k-ary malware can be used to infect systems by dividing malicious payloads into k parts [8]. It is imperative that we address these issues to make blockchains a safe technology to be used more widely.

1.3 Aim of study

To combat such vulnerabilities, we aim to create a blockchain based architecture where users can communicate over the network in a secure way so that no malicious entities can inject malware into the blockchain network by utilizing the properties of blockchains along with convolutional neural network (CNN) to aid in the detection of malwares. We also want to make sure that data transfer is done securely and no user node gets infected with malware through the blockchain. We further wanted to test out different CNN models with the blockchain architecture to find out which one worked best in terms of time and computational resource efficiency.

1.4 Research methodology

We have created a blockchain architecture and tested it with two different CNN models: the VGG-16 architecture and a customized model. For training and testing both the CNN models, we have used a secondary dataset which we have found to be used in multiple research papers. To test the combined blockchain and CNN architecture, we have used a mixture of primary dataset and secondary dataset. We first tested the performance of the custom model integrated with the blockchain and then tested VGG-16 in the same way. Lastly, we compare and analyze these two models to see which model works best overall.

1.5 Thesis outline

The rest of this paper is organized as follows:

Chapter 2 contains the literature review, which discusses previous work done in this field. It summarizes previously published relevant papers and briefly explains convolutional neural networks, blockchains, and algorithms developed on these so far.

In chapter 3, we have explained about our datasets and why we worked with secondary dataset. We have discussed the data for the CNN model and the data for the combined blockchain and CNN model separately.

Chapter 4 explains the proposed model and workflow for this research.

In chapter 5, we have explained how we implemented our proposed model.

Chapter 6 concludes and summarizes our work and discusses further research that can be done in this field.

Chapter 2

Related Work

2.1 Convolutional Neural Network

2.1.1 What is a convolutional neural network?

CNN is a profound neural network initially intended for picture investigation. CNN is structured by the association and usefulness of the visual cortex and intended to replicate the physicality of neurons inside the human cerebrum. Each set of neurons within the CNN are broken down into a 3D structure which is in turn assigned a small portion of the image. Similarly, each gathering of neurons has some expertise in distinguishing one piece of the picture. Convolutional operations are performed by the numerous filters of the convolutional layers. It steadily contains two fundamental activities, convolution and pooling. Convolutional layers are produced using a few element maps, and every unit of highlight maps is produced convolving a little area from the information feed which is known as the nearby responsive field. Pooling layers are generally utilized following convolutional layers which were created to rearrange the data and decrease the size of highlight maps. As such, in convolutional layers, pooling layers make a dense element map from each element map. Furthermore, these layers are also known as subsampling layers. The two most common cycles for pooling are max-pooling and average-pooling. A CNN compresses a fully connected network by reducing the number of connections and sharing weights on the edge. In addition to this, max-pooling further reduces complexity. For the training purpose, like MLP learning weights, CNNs will gain proficiency with the most ideal filters for perceiving explicit patterns and examples. However, CNN learns various filters in the process. An attribute is taught to each filter. As a result, in each layer, it learns various filters [9].

2.1.2 Work related to convolutional neural networks

Research on malware has shown that malwares can generally be classified into categories and this paper suggests the deep learning method of convolutional neural network (CNN) for this purpose. The approach is data independent where the first step is malware visualization as image. Malware binary file is converted into grayscale image which is used as the input to the CNN. A CNN which is a feed-forward neural network, consists of three layers: convolutional layers, pooling layers and fully connected layers. The output of the CNN is a set of scores for all the

different malware classes and the class which has the maximum score is chosen as the prediction for that particular malware class. The network is trained using the cross-entropy loss function where *.bytes* files were used for training. Two different settings were used to carry out the experiment which produced results with more than 98 percentage of accuracy [10].

The model used to classify malware proposed in this paper is inspired by the concept of tagging proposed by Yong *et al.* (2017). This model has four layers: the CNN layer, the GRU layer, the DNN layer as well as the sigmoid layer. There are three sub layers of the CNN layer: the first layer that is the convolution layer, is where the user defined or distributed uniformly filter is transformed with the input matrix to create a second matrix that is transferred to the activation layer. The ReLU, tanh, or sigmoid activation functions are added to the matrix in the activation layer and the effects are passed on to the pooling layer. For the production of the output value of CNN, max pooling or average pooling is applied in the third layer which is the pooling layer. The GRU layer and the DNN layer are composed of equal number of user defined GRU and DNN units. The GRU passes on the output to the correspondent DNN, where it is evaluated by weighing and summing and then sent to the activation functions. The output produced is in the form of a vector where the dimensions are the number of nodes of the GRU and DNN. Finally, all the DNN outputs are composed into a single output and passed onto the sigmoid function where it classifies the output. In the training stage, a truth value is given to the neural system. The difference between this and output classification is the error that can be used in backpropagation for training and changing weights. The dataset used for the experiment contains 21741 samples, out of which 10868 are for training and the remaining are for testing. The dataset contains 9 different families of malware. This model achieves an accuracy of 92.66 percentage, which is considered state-of-the-art but its accuracy can be further improved by putting in more effort [1].

Hamzeh (2019) implements a static malware detection approach that uses raw bytes to extract features and constructs a unique and parallel convolutional neural network architecture (CNN). Since the length of byte sequences in each file varies, the file header is known to be the parallel-CNN input, which is generally the first 1024 bytes of the file. In comparison, to construct a fixed and exclusive numerical vector for each byte, word embedding methods are implemented. Embedding approaches were designed to solve the problems of text classification; however, their methodology employs one of the most effective embedding techniques called word2vec by considering each byte as a word. The method that is proposed could be summarized in two steps: (1) generating numerical vectors for bytes using the word2vec model, (2) distribution to parallel-CNN of these vectors to execute highlight extraction and detection of malware. It is a static byte-level approach; therefore, it does not need to dismantle or extract the graph which is calling the function or finding API calls that are time-consuming and often vulnerable to error. They also presented CNN with a parallel framework capable of automatically extracting n-gram characteristics and reducing dimensions, so the complexity of finding a proficient algorithm for element extraction is removed [11].

In this paper, the authors state that more than one criteria is required to evaluate a

CNN model which will allow for the ability to deal with the imbalance of datasets. Therefore, they proposed a CNN model for the classification and identification of malware and non-dominated sorting genetic algorithm II (NSGA-II) to tackle the imbalance of datasets. For visualizing the malicious codes, they used the idea of converting executable binary files of malicious codes into grayscale images. The architecture of the neural network used in this paper consists of two layers of convolution, one layer of pooling and two dense layers. Using the Rectified Linear Unit (ReLU) activation function, the convolutional layers extract different features from the input images while max pooling is used for downsampling the features map in the pooling layer. The last layer utilizes softmax regression that changes over the yield of the forward proliferation of the neural system into a probability distribution. Due to the quantities of certain malware images being larger than others, an imbalanced dataset is created which is regarded as a multi objective optimization problem. Undersampling is thus used to eliminate some samples from the sample set and the NSGA-II algorithm is applied for the optimization of the sampling weights of different malware families. 1-TPR and FPR are the two objective functions found which were evaluated using the cross-entropy loss function and the Pareto optimal solution which met the evaluation criteria was chosen. An increase in the accuracy, recall rate, along with reduced loss and TPR was achieved when experiments using the proposed model were carried out. However, this paper proposed a method for dealing with small scale images which would not suffice in the real world where much larger resolution of images have to be dealt with [12].

2.2 Blockchain

2.2.1 What is a blockchain?

Since the inception of blockchain back in 2008 (first outlined in 1991), slowly yet gradually it is becoming the reliable name of security when it comes to sharing data online or keeping data tamperproof. In simple words, blockchain is actually a database that is shared through a network of computers. But unlike a typical database storing information, blockchains store data in blocks that are then chained together. It is really hard to alter when a record or information has been attached to the database or chain. To describe the functionality of blockchain in brief, securely storing the specific information in a block contains few of the steps. First of all, the data is checked by the network. The computers in the network are called ‘nodes’. These nodes examine the validity of the information. The data that have been approved by the network are then added into a block. A unique code called ‘hash’ is found in each block. It also holds the hash of the previous block in the chain. A hash code is usually generated by a math function that takes digital information and constructs a string of letters and numbers from it. Two of the important attributes of hash codes are: 1. Regardless original file’s size, a hash function always produce a code of same length. 2. A new hash key will be generated if there is any change in the original input. So if a manipulation does occur by the hackers, the next block in the chain still contains the old hash. So the hacker will have to recalculate this to restore the chain and then the next one and so forth. It will take an immense amount of computational power to recalculate those hashes, thus making the chances of the hack being successful rather impossible [13].

It is quite feasible to store various types of information on blockchain, but the most common till date has been a transaction ledger. A blockchain consists of two key components: a decentralized network that enables and verifies transactions, and the network's immutable ledger. This shared transaction ledger can be accessed by everyone inside the network, but there is no single point of failure from which documents or digital assets can be compromised or manipulated. There is also no organization controlling the data because of the decentralized trust. In the case of the cryptocurrency called Bitcoin, uses blockchain technology in a decentralized way such that no person or community has control—rather, all users maintain control collectively. Another vastly used blockchain technology is Ethereum. After Bitcoin, it is the second largest crypto-currency by market capitalization. Ethereum's feature includes open-source, decentralized blockchain with smart contract feature. Smart contract is basically a transaction protocol that is intended to automatically conduct, monitor or record legally appropriate events and actions according to the terms of a contract.

Blockchains can also be public and private. Public blockchain needs no permission. Inside the blockchain, anyone can access the network, read, write and participate. It's also decentralized and does not have any single entity which controls the network. Private blockchain on the other hand needs permission. Based on access control, private blockchain restrict individuals who might participate in the network. More than one entities control the network which leads to third parties to transact.

2.2.2 Work related to blockchain

Blockchain has been a revolution for the security of the Internet with it being decentralized within a peer to peer network. blockchain maintains a ledger mechanism where a copy is saved by all the members of the chain. When a transaction takes place, it is verified by other members by decrypting a hash function generated. Being decentralized and open to everyone in the chain, the trustlessness factor comes into play. The mechanism eliminates two crucial security threats namely: Double Spending and Record Hacking. As every transaction generates a hash function which needs to be validated by miners in a peer to peer network, it prevents multiple transactions (Double Spending) to be made. Furthermore, as everyone owns a copy of every single transaction made, it is impossible to alter the blocks of transaction (Record Hacking). But blockchain is far from being immune to other cybercrime due to the fact that each node as an entity holds the possibility to be hacked and manipulated otherwise. Therefore, further enhancements in the technology has been suggested which can better help to combat cyber threats [14].

A paper was suggested to deal with malware identification, an engineering that continuously stores and transmits the attack signatures safely with the ultimate goal of prompt detection. It is a standard format for the storage and distribution of signature-based IDS attack signatures to support nodes running various IDS by constructing a blockchain-based private and public engineering. The blockchain architecture enables public nodes to securely enter and evaluate stored attack signatures in real time without permission. It is developed using the Ethereum blockchain platform to extract, convert, store and distribute cyberattack signatures using both public and private blockchain features. The proposed architecture is divided into 3

phases in which only approved nodes that detect attacks can extract the signature. Transactions that are submitted and the privilege of the owner are authenticated; the signature is transformed to standard format and checked in the storage of signatures. Smart contract manages both authentication and signature conversion, while validation is done by blockchain consensus protocol. Transaction and owner verification ensures that transactions are not submitted by unauthorized nodes. They created a script that converts signatures from one IDS to a common format compatible with another for signature format development. After effective conversion to standard format, the pending transaction is installed into a block by an accepted node. The block is submitted for validation to the blockchain network. The transaction address is given to the owner after a new block has been chained to the blockchain (sender). The current state of the blockchain (i.e., the update of a new block) is communicated to each node within the blockchain network. A duplicate of the update is obtained by each node (approved and unapproved) in the blockchain network. Blockchain nodes ask for transaction addresses and ABI to download attack signatures mined by other consortium members into the blockchain from the database [15].

A new framework namely Consortium Blockchain is introduced which combines the idea of both a public and a private blockchain. This helps in giving a better control for the trustlessness problem even with the distributed peer-to-peer network. The workflow suggests that feature extraction can be done in order to verify malicious entities. For the approach, two separate methodologies are taken into consideration namely, static and dynamic feature analysis. The static feature analysis method extracts the physical attributes of an application and cross checks against samples in a block chain. The dynamic feature analysis method works with the performance measures of an application. This verification is done by a semi decentralized (consortium) blockchain which validates the transactions of the global (public) blockchain. Any transaction taking place within the framework, takes a mixed route, eliminating the risks of an intervention trying to manipulate the data. A further breakdown of the model can be sorted into four layers: network, storage, support and application. The network layer being responsible for connecting and synchronizing the nodes. The storage layer acts as a database holding the entries of any malicious entities, which are used in the cross check. The application layer is the user interface and the support layer is the bridge connecting the application layer to the storage layer with managing key aspects for any transaction [16].

A further enhanced approach to the Consortium Blockchain theory is made in this paper by taking two private blockchains, one being internal and the other being external. The model has been experimented for mobile applications in the app store. The internal private blockchain holds the Static and the Dynamic feature extractors. The static feature extractor helps in obtaining the physical analysis of an application such as library, package and permissions while the dynamic feature extractor works on dedicated performance analysis of the applications behavior such as CPU and memory usage. The external private block chain works with detection engines. These engines are responsible for scanning hidden applications using machine learning techniques and tokenization. Finally, the results are appended to the consortium block chain which helps in determining the authenticity of the application [1].

Jawad *et al.* (2020) presents a behavior capturing and confirmation methodology in blockchain upheld smart-IoT frameworks that can have the option to show the trust-level confidence to outside networks. They characterized a custom Behavior Monitor and implemented a chosen node that can remove the action of every device and break down the behavior utilizing deep machine learning strategy. They also focused on applying a filter on sensor-level that can stabilize output from single/multiple sensors to avoid faulty or malicious sensors in the network. Furthermore, they conveyed Trusted Execution Technology (TEE) which can be utilized to give a protected execution environment (enclave) for sensitive application code and data on the blockchain. The first phase of deployment signifies that a single device from each zone is designated as a Main or Master node, which can be considered as a certification authority. Hardware Model of IoT The hardware design they worked with in their proposed system for prototyping comprises multiple raspberry pi's. The main/master node is configured on raspberry pi-3 for the sake of more resources. In order to configure blockchain a local private blockchain is deployed on a master node (Raspberry pi-3) of each zone and populated with the hashes of transactions generated from smart- devices. The main achievement in this research is to define a behavior monitor that can classify the behavior of the devices and compute a level-of-trust for each zone. As referenced before, all the nodes (followers) in a particular zone try to do their tasks (read, compose) by means of the master/main node. The procedure of behavior identification and observing comprises the stages: Data assortment, Feature extraction, Training model, Continuous Behavior Monitoring [17].

2.3 Blockchain with Neural Networks

2.3.1 A combined approach

Quite a bit of work has been put into combining the two technologies to build secure systems. Blockchains and neural networks have been gaining a lot of attention over the past few years. However, combining these two is a relatively new idea. Nevertheless, this combination has shown a lot of potential. This section discusses a few such works.

2.3.2 Work that combines both the technologies

A new architecture which they call the “DeepRing” architecture has been proposed. Its purpose is to protect a DNN i.e., an AI system from malicious attacks and adversaries and it combines the architecture of a CNN along with the security centered features of a blockchain such as cryptographic encryption at each progression, transitive hash, and its decentralized nature. The architecture is mainly inspired by a blockchain and the only difference from a blockchain is that the DeepRing is made up of a finite number of blocks forming a ring as opposed to a blockchain’s infinite, ever growing number of blocks. Each block represents a layer of the DNN and does the following: stores parameters of the layer, computes output of the layer, updates the ledger after output computation, furthermore, approves output of the following layer. As mentioned before, this architecture represents a ring, and the starting

and ending point of the ring is called the Ouroboros block. This block does not correspond to any layer of the DNN but has a lot of important functions and is the focal point for detecting attacks. They have tested this architecture using two types of attacks: by tampering with the parameters of the most important layers in the network or by perturbing the inputs to the network. For tampering with the parameters, they have proposed an algorithm. They have modeled VGG-19 architecture proposed by Simonyan and Zisserman (2014) trained on MNIST2, CIFAR-10, and Tiny-ImageNet3 in a Deep-Ring framework. They have additionally viewed a neural network with five thick layers with the accompanying properties: Number of nodes in each layer [900, 600, 300, 100, number of classes] and ReLU activation for all layers except the output layer. Softmax activation is used for the output layer. They have trained this network on MNIST and CIFAR-10 datasets. Then they have experimented on the architecture in the aforementioned two ways. Results show that When the original model is used for recognition, the VGG-19 model 10 yields 99.07 percentage, 83.89 percentage, and 76.12 percentage object acknowledgment precision on MNIST, CIFAR-10, and Tiny ImageNet databases respectively. When parameters or inputs are tampered with and no security measures are provided, the accuracy drops by 20.71 percentage, 47 percentage and 34 percentage on CIFAR-10, MNIST and Tiny ImageNet respectively. In this way, the proposed DeepRing engineering is without flaw in light of multiple verification blocks, for example, approval/consensus and hash functions. In the future, they have plans to work on this architecture to make it more efficient in terms of computational complexity and to defend against input image perturbation [18].

This paper proposes a deep learning-based blockchain framework to surpass the security challenges of a centralized software defined industrial network such as single point of failure and distributed denial of service (DDOS) attacks. A BlockSDSec model is designed along with a DDOSattack model. The BlockSDSec model consists of four layers namely, the Application Layer, the Control Layer, the forwarding/Blockchain Layer and the Host Layer. Since the proposed model segregates the security and services part of software defined network (SDN) from management and control, the Application Layer constitutes of all the security applications. The Control Layer is responsible for the registration and verification of any new switches or devices using zero proof of knowledge (ZKP) algorithm. Moreover, the validation of incoming packets from existing switches in the blockchain which request for a flow table update is also performed in this layer. By forwarding the request of flow table entry to a deep boltzmann machine (DBM) anomaly detector algorithm, the validity of the switch request is checked. The DBM flow analyzer identifies different attack patterns by working on features such as, Percentage of pair-flows (PPF), growth of single-flows (GSF), growth of different-ports (GDP) etc. Comprising the OpenFlow switches which are connected to each other forming a private blockchain is the Forwarding/Blockchain Layer. A new block is added to the chain only after it passes the validation phase and receives a consensus. The results of the experiment carried out using the proposed model show a 5 to 10 percent better performance in accuracy compared to existing models [4].

Chapter 3

Data collection and preprocessing

There are two methods of data collection: Primary data collection and Secondary data collection. For the CNN model, we have mainly used secondary data and for the proposed model, we have used a mixture of primary data and secondary data: the malware dataset is secondary while the non malware dataset is primary.

3.1 Dataset for training and testing the CNN model

For training and testing of the CNN model, we have used the Malware Image dataset from Vision Research Lab (Maling) [19]. The Maling dataset consists of grayscale images belonging to 25 malware families, with the total number of malware images being 9,339. We have used around 6537 images as the training set and 2802 images as the test set. This dataset already contained the grayscale images which were classified into 25 different malware families.

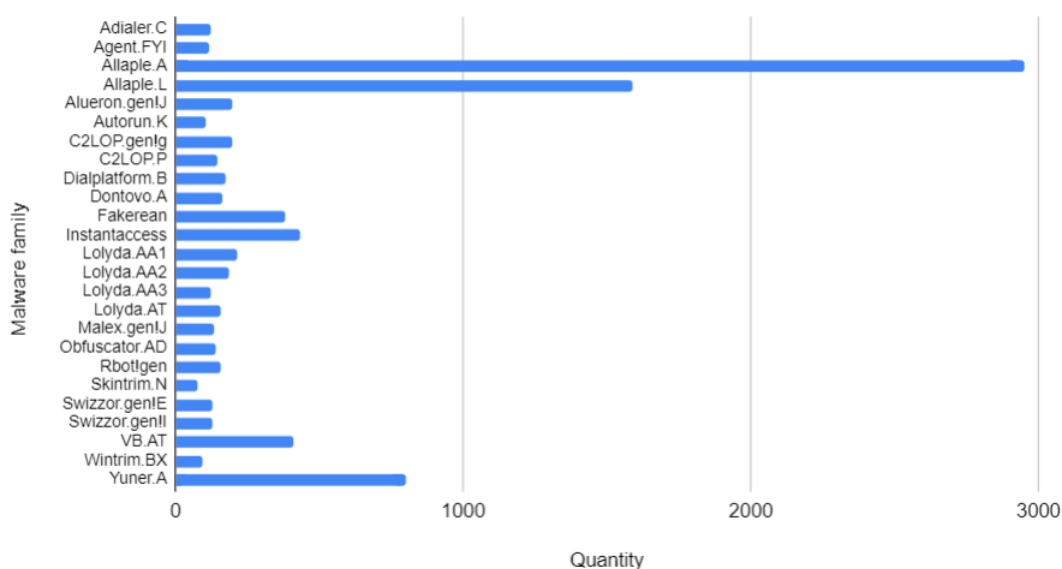


Figure 3.1: Quantity of Malware Images in Different Families

3.2 Dataset for testing the model

3.2.1 Data collection

For testing the model, we have used a mix of malware and non-malware data. This dataset contains a total of 350 files, of which 250 are malware and 100 are non-malware. The malware part of the dataset is from the Microsoft malware dataset [20] which contains *.bytes* files from 9 different families of malware which are:

1. Ramnit
2. Lollipop
3. Kelihos_ver3
4. Vundo
5. Simda
6. Tracur
7. Kelihos_ver1
8. Obfuscator.ACY
9. Gatak

Originally the Microsoft dataset consisted of 21741 samples out of which 10868 are training samples and 10873 are testing samples. There are two kinds of files in it - *.bytes* files and *.asm* files. The raw data for each file are hexadecimal representations of the file's binary content. Since we have collected the data from Microsoft Malware Classification Challenge, the dataset provided is not classified into their respective malware families. With the help of the csv file composed of the class labels associated with the train set, we chose 250 *.bytes* files comprising approximately 28 samples from each of the 9 malware families.

The non-malware part consists of files from our local machine, which includes 28 pdf files, 28 powerpoint files, 20 images, 15 doc files and 9 txt files.

Chapter 4

Proposed Model

We wanted to create a blockchain model such that it would not allow users to upload any malicious content to the blockchain during the transaction process. To accomplish this we thought of combining Convolutional Neural Network with the blockchain architecture. We have used CNN to identify malware by visualizing files as grayscale images. Image representation of malwares allow us to detect any small changes made in a file by potential attackers [10]. Two different CNN models have been used to test which kind of architecture works better when integrated with a blockchain model. In our combined model we have used a private blockchain architecture. The details of the models used are described in this chapter.

4.1 CNN Layers and Functions

4.1.1 Convolution layer

In the convolutional layer, one matrix comprises a set of learnable parameters called kernel, and the other matrix acts as the restricted portion of the receptive field and the output is dot product of these matrices. Despite the size of the kernel being smaller than an image, it allows a more in depth sparse interaction. The dimensions generated by the kernel are proportional to the height and width of the image. This two-dimensional representation, referred to as an activation or map of features, provides the response of the kernels for each spatial location of the image, and the sliding size is referred to as the kernel stride [21]. Padding is used when the filter does not fit the input image perfectly and we want to avoid losing pixels on the perimeter of the image for which the extra pixels are generally set to 0 [22].

4.1.2 Pooling layer

Pooling layers serve the purpose of reducing the sensitivity of convolutional layers to location and of spatially downsampling the representation [23]. While pooling downsamples the image in terms of height and width, the number of channels (depth) remains the same [24]. There are different kinds of spatial pooling functions such as max pooling, average pooling and sum pooling. We have used max pooling which works with the largest component from the rectified feature map [25].

4.1.3 Dropout layer

To prevent a model from overfitting, dropout layers are used which operate by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each training phase change [26].

4.1.4 Flatten layer

To transform the data into a one dimensional array as input for the fully connected layer, we use the flatten layer. A single long feature vector is the result of flattening the output of the previous layers [27].

4.1.5 Fully connected layer

Fully connected layers are feed forward neural networks where all the neurons are fully connected to all the neurons in the preceding and subsequent layers [24], [21].

4.1.6 ReLU activation function

Rectified linear activation function is a piecewise linear function which provides the activation sum input with more sensitivity and prevents fast saturation. The activation function used by rectified linear units is $g(z) = \max\{0, z\}$. Since rectified linear units are nearly linear, they retain many of the properties of linear activation functions which allow for easier optimization with gradient-based methods and generalisation while acting as a nonlinear function which always outputs negative values as 0. Usage of ReLU activation function in hidden layers improves the overall performance of the network [28], [29].

4.1.7 Sigmoid activation function

Sigmoid activation function or logistic function is a nonlinear activation function which is S-shaped. The input to the function is translated to a value between 0 and 1. This property makes sigmoid activation function be useful when used in the output layer of the network for predicting the probability as an output. This property of the sigmoid function, makes it a great candidate to be used in our CNN models [30], [29].

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

4.1.8 Cross entropy loss function

Loss functions are used to minimize errors in the network during the training process. To measure the performance of a model whose output is a probability value between 0 and 1, cross-entropy loss, or log loss is used [31], [32].

4.2 The custom CNN model

In our first model we have used five layers, namely two convolution layers and three dense layers. We divide the Maling dataset into 80 percent train set and 20 percent test set ratio. The input to the first convolution layer is a grayscale image of fixed size 64 x 64. In the first convolution layer we used 30 filters of 3 x 3 kernel size and in the second layer of convolution we used 15 filters of 3 x 3 kernel size with default stride. Next, we used a pool size of 2 x 2 in the max pooling layers. Default strides are used for both convolution and max pooling layers. The first dropout layer drops 25 percent neurons whereas the second dropout layer drops 50 percent of the neurons. 128 neurons are applied in the first dense layer and for the second dense layer 50 neurons are used. The two convolution and dense layers use the ReLU activation function. However, the last dense layer uses the sigmoid activation function to classify whether an image belongs to any of the malware classes or not. We have used the cross entropy loss function with Adam optimizer.

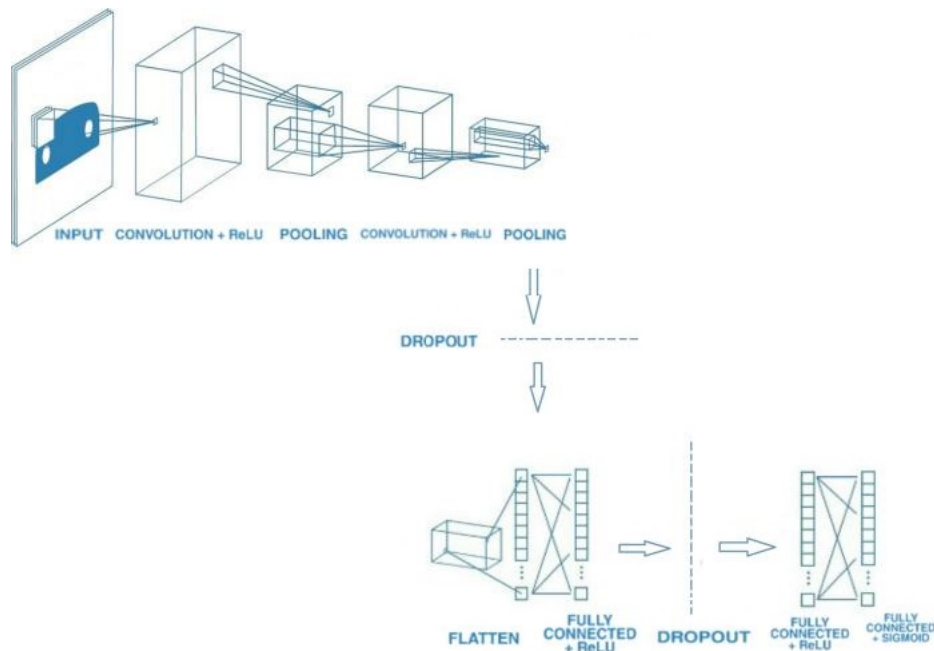


Figure 4.1: The architecture of the custom CNN model

4.3 VGG-16

For our second model, we have used the VGG-16 model which consists of 16 layers, namely thirteen convolutional layers and three dense layers. The VGG-16 is a convolutional neural network architecture presented by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition” [33]. This framework has 16 layers that have weights, hence the name VGG-16. This network has approximately 138 million parameters. We divide the Maling dataset into 70 percent train set and 30 percent test set ratio. We used a stack of convolutional layers followed by the max pooling layer, doubling the filter size after each max pooling layer. This helps to better

analyze the feature map. The input in this model is taken as a grayscale image of size 224×224 . In the first layer we used 64 filters of 3×3 kernel size with ReLU activation keeping the padding the same. Next we used another Convolution layer with similar stats. We then used a max pooling layer of pool size 2×2 and stride 2×2 . After that we run two layers of conv2D with 128 filters. We then again add another max pooling layer with the pool size of 2×2 . After that we add three more convolutional layers with 256 filters followed by another max pooling layer. We then add two sets of three convolutional layers with 512 filters along with the max pooling layer. After that we add the flatten layer accompanied by two dense layers with 4096 neurons with ReLU activation. Finally we add another dense layer which has the same number of neurons as the number of classes in the malware dataset. We use a sigmoid activation in this layer to determine whether the input belongs to any classes of the malware dataset. Cross entropy loss function with RMSprop optimizer is used for optimization of the model. Fig 4.2 shows the architecture of the VGG-16 model [34].

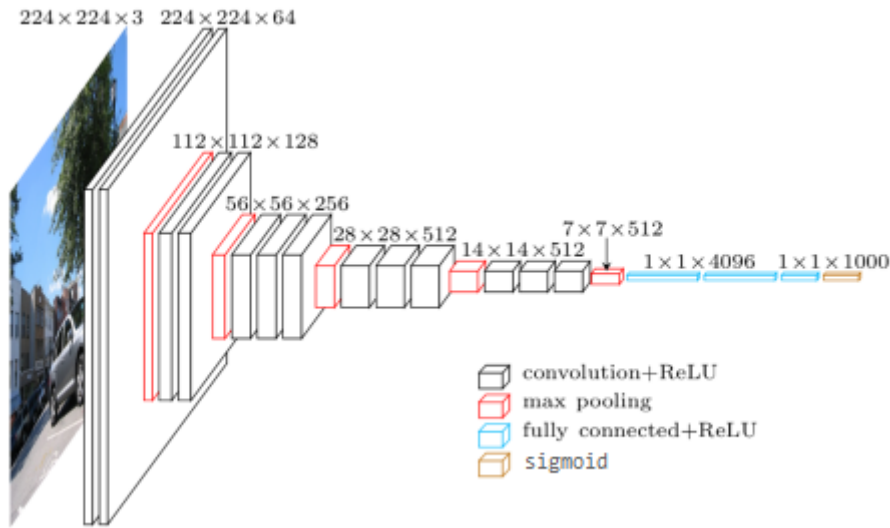


Figure 4.2: The architecture of the VGG-16 model

4.4 Blockchain

Blockchain is one kind of a database where information is grouped together and stored in blocks. They are distributed, decentralized peer-to-peer networks with a public ledger. Each newly created block is chained to the previous block through cryptography, which keeps the transactions' confidentiality intact, establishing a chain of data called the Blockchain [35], [36]. There are various types of blockchain networks - public, private, permissioned and consortium blockchains [37].

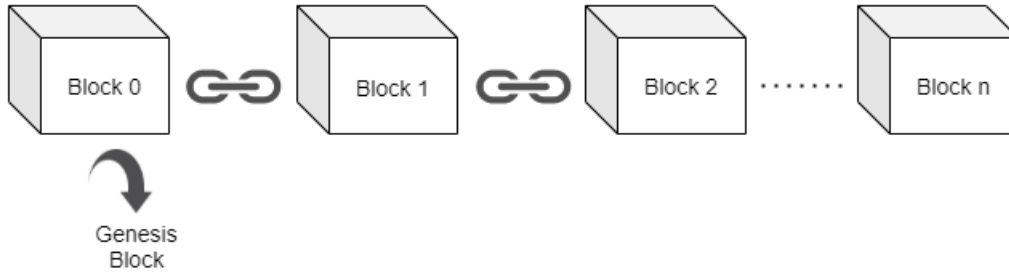


Figure 4.3: Basic architecture of a blockchain

Public Blockchain: Public blockchains are permissionless which means anyone can be a part of the blockchain network and participate in it. It is a decentralized system where a single entity does not have control over the entire network [36].

Private Blockchain: Private blockchains are more centralized and are permissioned. There are restrictions on the people who can participate in it with the use of access controls. One or more entities are responsible for governing the entire network [36].

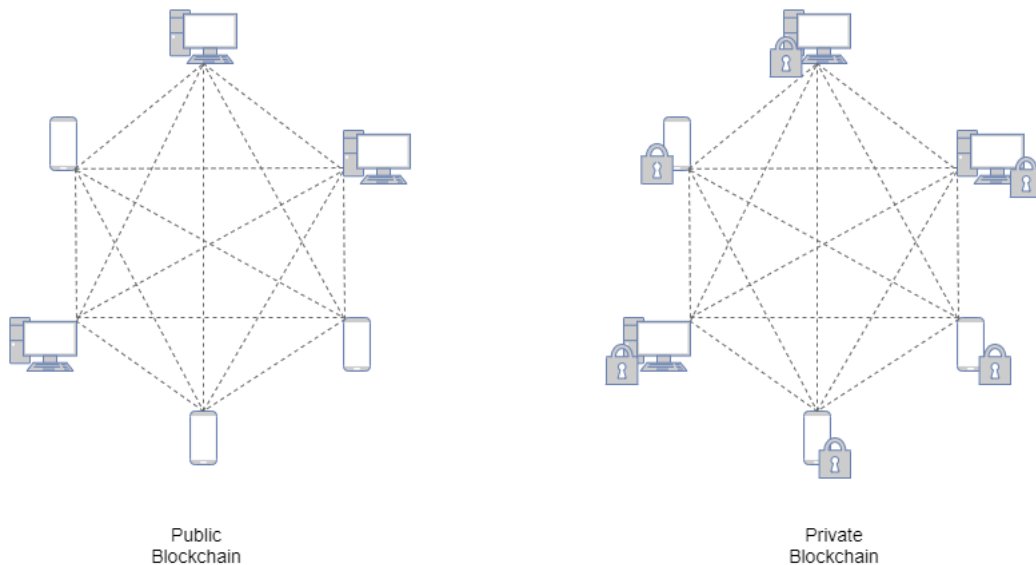


Figure 4.4: Topology of Private and Public blockchain

For our model we have used a private blockchain architecture in which a central organization has control over the network. Due to the lower number of authorized participants in private blockchains, the transactions per second are lower and the network is thus easily scalable. A private blockchain also consumes significantly less resources in terms of energy and power to function [36].

4.4.1 Genesis Block

The genesis block is the first block of any blockchain. It is also called block zero and it acts as a prototype for all the other blocks in a blockchain. Every block in a

blockchain stores the hash value of the previous block. However, since the genesis block is the first block, it does not store any such value, which means its “previous hash value” is zero. For our architecture, the genesis block contains a dummy string of data along with the hash and timestamp. The genesis block is automatically created when our blockchain is initiated [38].

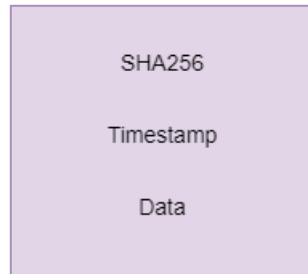


Figure 4.5: Architecture of a genesis block

4.4.2 Data Block

All the other blocks in the blockchain, besides the genesis block, are called data blocks. These are the main storage units in a blockchain. Whenever a transaction is made, it is recorded in these blocks. In our blockchain architecture, each data block contains information uploaded by the user, namely the sender address, receiver address as well as the hashes of the previous and current blocks, the timestamp and nonce. The blocks are added in a linear fashion with the help of a chain similar to a linked list [39].

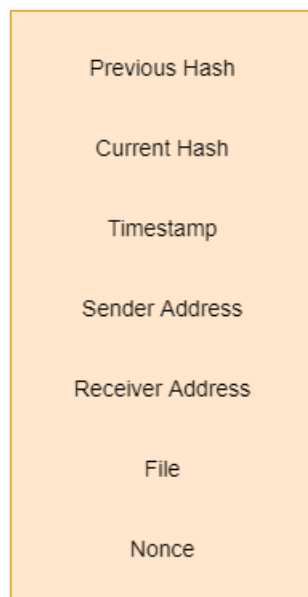


Figure 4.6: Architecture of a data block

The components in each of the blocks are described in details below-

Previous hash: Each block in our blockchain architecture is connected via the previous hash. The usage of cryptographic hash functions in the creation of the blocks makes blockchains resistant to data modification. The hash function used in our model is SHA256. Known to be immutable, the SHA256 algorithm is irreversible and thus maintains data integrity [40]. In our blockchain architecture, all the blocks are linked to their previous block and therefore to the genesis block.

Current hash: The functionalities of hash functions, in our case the SHA256 hash function, allows the unique identification of the blocks in a blockchain network which is linked cryptographically [41]. By using the sender address, receiver address, timestamp and nonce, the hash function is generated in our blockchain architecture.

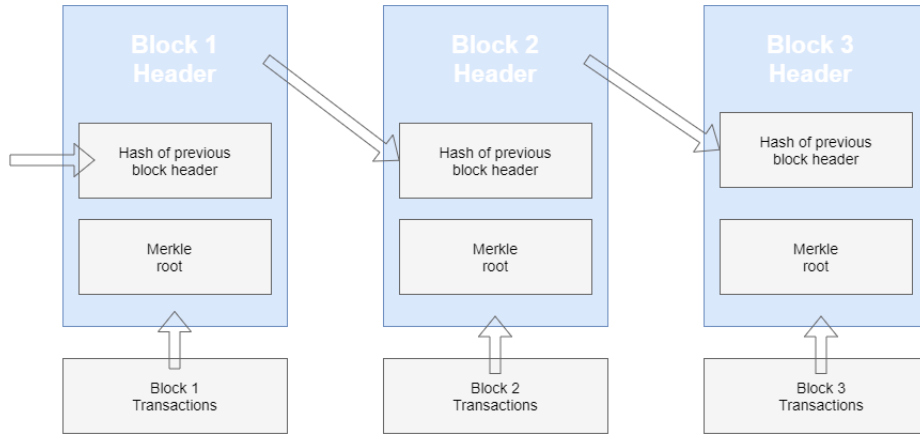


Figure 4.7: How the previous hash and current hash works

Timestamp: When a block is mined, the timestamp for it is saved which tells us when this block was mined. This makes every block in a system unique and immune to mutation by third parties as this timestamp is also used to create a unique hash for the block.

Nonce: Nonce stands for “number used once”. It is a random whole number that is added to a hashed block in a blockchain. This is the number that miners solve for, that is, adjust its 32 bit field to turn it into a valid number which can then be used to hash the value of a block. In our blockchain architecture rather than randomly generating numbers which could have a possibility of repetition, we incremented the nonce for each block. This nonce along with timestamp and user information is used for creating the hash [42], [43].

4.5 Combined model

When a user fills up the form on the interface with the sender address, receiver address and the file, these information are stored in the database. The file is also uploaded in a folder on the server. A python script on the server which runs the Flask API, takes the file from the folder and converts it into a grayscale image. It then loads the CNN model and evaluates the input against the model. The output of the CNN model is a nested list which contains either 1 or 0 depending on the malware class. The sigmoid function classifies the test input with the highest probability of similarity for each class of malware. If the similarity threshold exceeds 0.5, it gives an output of 1 respective to that class. This way the script can determine whether the output contains any positive values which the script then can classify as a malware. If not, the file is considered to be non-malicious. The output is then passed along to the browser through a HTTP request via the Flask API. At the same time the blockchain model is triggered in the back-end. The main method in the blockchain model calls upon the Flask API running in the server with the help of Axios get request. If the response from the HTTP request is classified as a malware, the main method exits the script and the block is never made. However, if the response is classified to be a non-malware, then the main method creates a new transaction with the user inputs. A new block is thus created with the help of the SHA256 algorithm which generates a hash function using the input values: sender address and receiver address, timestamp, nonce and the previous hash. Once the block is created, it is then put in a queue to be mined. This hash value is also stored in the database to keep track of the user file. The server node is then responsible for mining the block. A predetermined difficulty level is set which offers optimal run times and also avoids the risk of getting flooded with block generations. Once the block is mined, it is then appended to the chain which marks a secure completion of the transaction. The workflow of the model is shown in Fig 4.8.

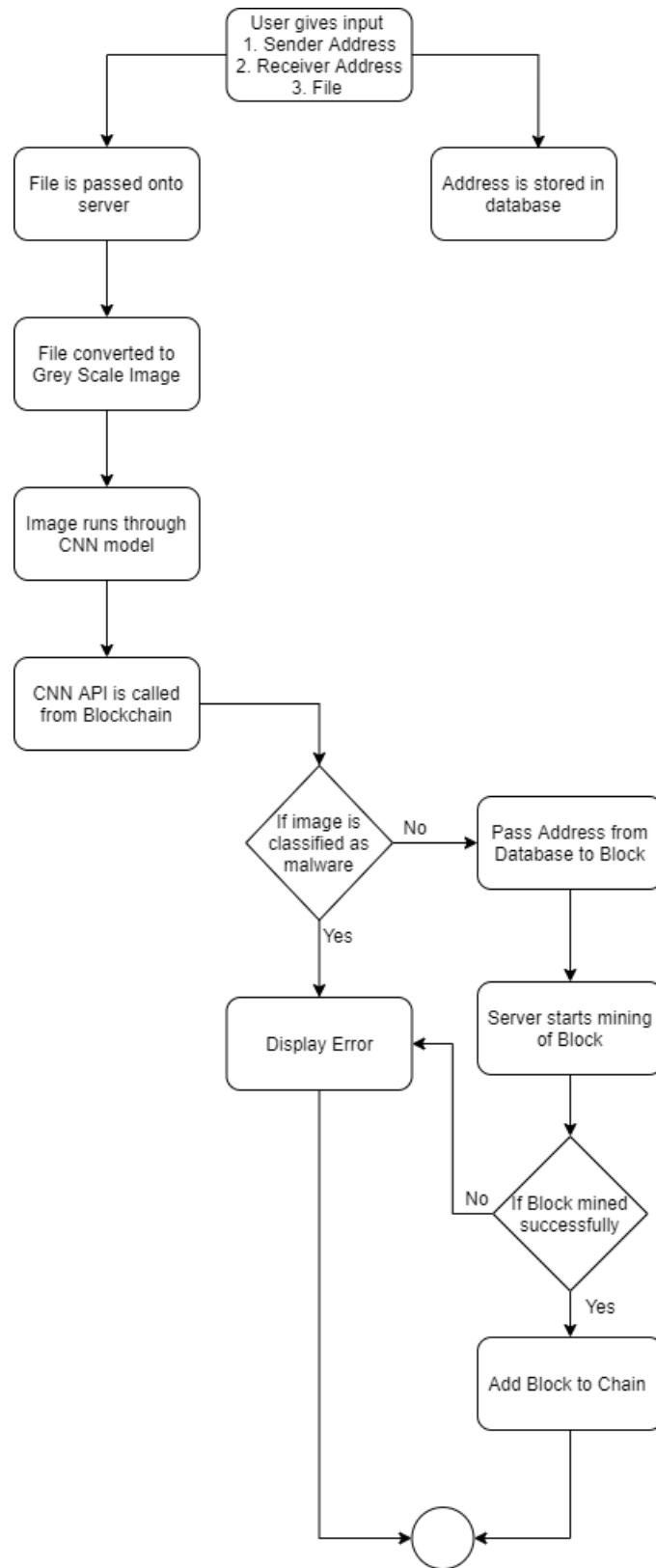


Figure 4.8: Workflow diagram of the model

Chapter 5

Experimental Setup and implementation

5.1 Machine specifications for implementation

For implementing and testing of all the models, we have used:

- CPU: AMD Ryzen 7 3700X
- GPU: AMD RX 5700 8GB
- RAM: 16 GB 2666 MHz

5.2 CNN

We have used Python programming language (Python 3) and Spyder IDE to create our CNN model. To set up the environment for the CNN model we have used the Keras library as well as the scikit-learn library. For our custom CNN model we have trained it for 15 epochs while for the VGG-16 model we have trained it for 5 epochs. For the custom CNN model, we used the default 234 steps per epoch and 10 validation steps and for the fully connected layer we used 128 neurons. For the VGG-16 CNN model, we used 70 steps per epoch and 5 validation steps to get a stable performance and for the fully connected layer we used 4096 neurons.

5.3 Flask API

A web API using Flask is created to assist the Blockchain in communicating with the CNN model. Flask is a web framework for python which allows users to build a web application by providing the necessary tools and libraries including the management of HTTP requests. Flask works by mapping HTTP requests or URLs to python functions, also known as routing. JSON (JavaScript Object Notation) format is used to return the data through the web API. Therefore, we chose Flask to host the CNN model in our local machine. To create a custom API for our model we first installed Flask and then imported the Flask library. As we are using multiple languages, we installed Flask-Cors and imported the CORS library which allows cross origin resource sharing. Flask helps to create a local server which shows the output of the CNN model on `http:127.0.0.1:5000` at all times.

```

app = Flask(__name__)
CORS(app)
cors = CORS(app, resources={
    r"/*": {
        "origins": "*"
    }
})

@app.route("/")
def get():

    return value

if __name__ == '__main__':
    app.run()

```

Figure 5.1: Flask implementation

5.4 File Conversion

To convert the user files into grayscale images, we first read the file as raw binary input. Then the length of the data is put in a numpy array. The data is converted to a vector and the data length array is taken in to be the shape of a square which is then padded with zeros. Next, we reshape the arrays with the squared padded length. Using the OpenCV library, we convert it to an image to be processed by the CNN model.

```

#file conversion to grey-scale img.

input_file_name = './uploads/test.bytes';
with open(input_file_name, 'rb') as binary_file:
    data = binary_file.read()
data_len = len(data)
d = np.frombuffer(data, dtype=np.uint8)
sqrt_len = int(ceil(sqrt(data_len)))
new_len = sqrt_len*sqrt_len
pad_len = new_len - data_len
padded_d = np.hstack((d, np.zeros(pad_len, np.uint8)))
im = np.reshape(padded_d, (sqrt_len, sqrt_len))
cv2.imwrite('./uploads/test.png', im)

```

Figure 5.2: File conversion

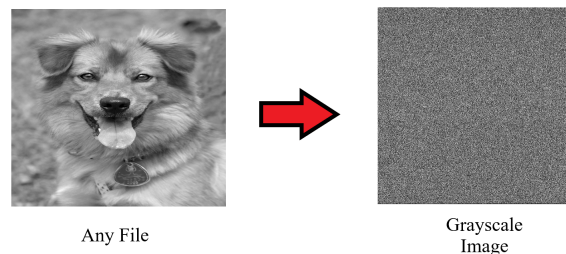


Figure 5.3: Any file conversion to grayscale image

5.5 Blockchain

Using JavaScript we have built a blockchain model which runs on Node.js. We have used the Crypto-js library and the SHA256 module that helps in the calculation of the hash function. For creating the blockchain model, we used the following classes:

Transaction class: This class takes the sender address, receiver address and file to create a transaction.

Block class: The block class takes the transaction, timestamp and hash of the previous block to create a new block.

Blockchain class: This class is responsible for creating the genesis block, mining of the pending transactions and adding a new block to the blockchain.

The functions implemented in the classes are explained below:

Create Genesis Block function: Creates the first block of the chain.

```
FUNCTION createGenesisBlock():  
  RETURN class Block(time.now, ‘‘genesis block’’);
```

Get Latest Block function: Calls the currently mined block from the chain.

```
FUNCTION getLatestBlock():  
  RETURN block from chain.length - 1;
```

Mine Pending Transaction function: This function calls upon the pending list of blocks, initiates the mineBlock function and passes the difficulty setting and then appends the mined block to the chain.

```
FUNCTION minePendingTransaction():  
  NEW instance Block = class Block(date.now(), pendingTransactions;  
  block.mineBlock(difficulty);  
  chain.push(Block);
```

Create Transaction function: This function is responsible for creating an un-mined block. It imports the data input from the user and creates an instance of the transaction and pushes it to the pending transactions list.

```
FUNCTION createTransaction() –  
  NEW instance transaction;  
  pendingTransaction.push(transaction);
```

Mine Block function: This function is responsible for the mining of the pending list of transactions. This function calls the calculateHash() and updates the current hash of the block.

```
FUNCTION mineBlock():  
  WHILE (hash.substring !== array.difficulty)  
  {  
    hash = calculateHash();  
    nonce ++;  
  }
```

Calculate Hash function: This function is used to generate the hash of the block. It imports the SHA256 library and all the necessary data of the input and returns the hash function in a string.

```
FUNCTION calculateHash()  
  RETURN SHA256(index+previousHash+timestamp+data+nonce).toString();
```

5.6 Combined model

For our final model we first created a user interface using HTML and PHP to allow users to upload a file onto the blockchain. The inputs from the users are inserted into an SQL database and the user files are uploaded to a local folder in the server. In the server, we keep a python script running by implementing a Flask API, which classifies the file by running against the CNN model and passes the result through a HTTP request. We bundle the blockchain code in a javascript file which is called from the PHP. We used the webpack module to wrap the javascript as it offers easier calling of javascript modules in the web browser. The blockchain javascripts calls the HTTP request from the Flask API and depending on the result, the block creation and mining process is either initiated or skipped. Fig 5.6 shows the form where user provides information.



Figure 5.4: User interface

Chapter 6

Result analysis

6.1 Results

For training and testing the custom CNN model it took approximately 3 minutes to run 15 epochs. When this model was saved to be loaded from the python script it had a total size of 4.5 megabytes. On the other hand, it took around 45 minutes to run 5 epochs for training and testing the VGG-16 model and the size of the saved model is 1 gigabytes. Since the custom CNN model had significantly fewer layers and thus neurons in comparison to the VGG-16 model, in addition to the training and testing time being less, the size of the saved file was also smaller. After training and testing both the models, the custom CNN model had an accuracy of 95.8% and the VGG-16 model had an accuracy of 94.9%.

```
Epoch 10/15
234/234 [=====] - 11s 46ms/step - loss: 0.1568 - accuracy: 0.9467 - val_loss: 0.1468 - val_accuracy:
0.9593
Epoch 11/15
234/234 [=====] - 11s 46ms/step - loss: 0.1603 - accuracy: 0.9485 - val_loss: 0.1329 - val_accuracy:
0.9647
Epoch 12/15
234/234 [=====] - 11s 46ms/step - loss: 0.1415 - accuracy: 0.9517 - val_loss: 0.1243 - val_accuracy:
0.9652
Epoch 13/15
234/234 [=====] - 11s 46ms/step - loss: 0.1296 - accuracy: 0.9558 - val_loss: 0.1422 - val_accuracy:
0.9599
Epoch 14/15
234/234 [=====] - 11s 46ms/step - loss: 0.1373 - accuracy: 0.9569 - val_loss: 0.1434 - val_accuracy:
0.9582
Epoch 15/15
234/234 [=====] - 11s 46ms/step - loss: 0.1198 - accuracy: 0.9584 - val_loss: 0.1240 - val_accuracy:
0.9663
```

Figure 6.1: Custom CNN Model Results

```
Epoch 1/5
70/70 [=====] - 559s 8s/step - loss: 0.8784 - accuracy: 0.5294 - val_loss: 1.0211 - val_accuracy: 0.6450
Epoch 2/5
70/70 [=====] - 568s 8s/step - loss: 0.7745 - accuracy: 0.5783 - val_loss: 0.5842 - val_accuracy: 0.6350
Epoch 3/5
70/70 [=====] - 555s 8s/step - loss: 0.6942 - accuracy: 0.6539 - val_loss: 0.3795 - val_accuracy: 0.6900
Epoch 4/5
70/70 [=====] - 549s 8s/step - loss: 0.3848 - accuracy: 0.8544 - val_loss: 0.0034 - val_accuracy: 1.0000
Epoch 5/5
70/70 [=====] - 549s 8s/step - loss: 0.4510 - accuracy: 0.9494 - val_loss: 6.5754e-04 - val_accuracy:
1.0000
```

Figure 6.2: VGG-16 CNN Model Results

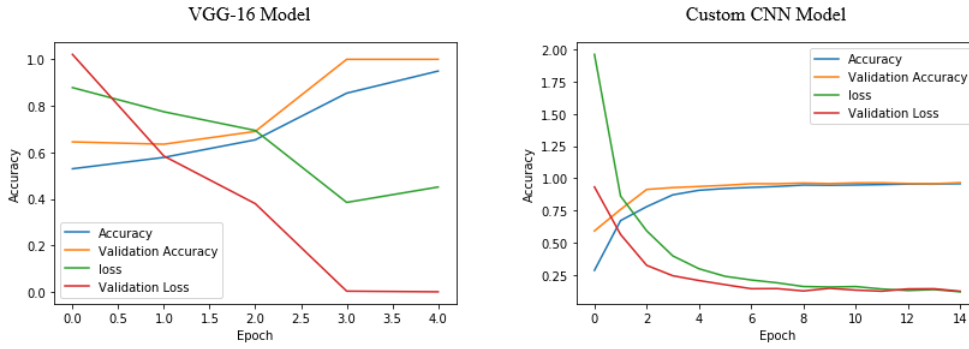


Figure 6.3: Accuracy and loss comparison between the two CNN models

After testing our blockchain architecture with both of the CNN models we saw that the longest time required by the custom CNN model to get a HTTP server response was 0.5 seconds while the shortest time recorded was 0.3 seconds. The longest time needed to get a HTTP server response by the VGG-16 model was 24.6 seconds and the shortest time was 6.2 seconds. Again, due to the VGG-16 model having a larger saved file size which has to be loaded every time a user gives an input, the delay experienced is much longer than the custom CNN model.

Model	Shortest time(s)	Longest time(s)
Custom CNN model	0.3	0.5
VGG-16	6.2	24.6

Table 6.1: Time required to get server response

As we had tested our entire architecture using the Microsoft dataset [20] which had malwares from completely different families than the ones which we had used to train and test our individual CNN models, the results show that both the CNN models used were able to detect malwares belonging to various classes quite well. While the custom model had an accuracy of 85.7% and 89.27%, the VGG-16 model had an accuracy of 77.1% and 92.3% when classifying non-malware files and malware files respectively.

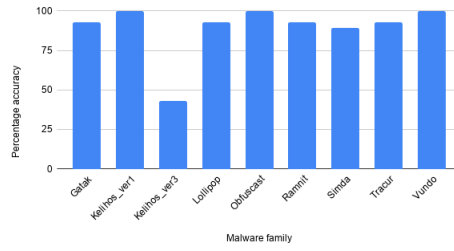


Figure 6.4: Percentage accuracy of custom model in classifying malware files from different malware families

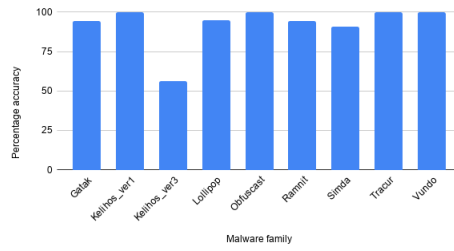


Figure 6.5: Percentage accuracy of VGG-16 in classifying malware files from different malware families

Overall, in terms of accuracy in correctly identifying malware files from non-malware files, we have discovered that the custom CNN model can do so 88.9% of the time and the VGG-16 model is able to do it 90.3% of the time.

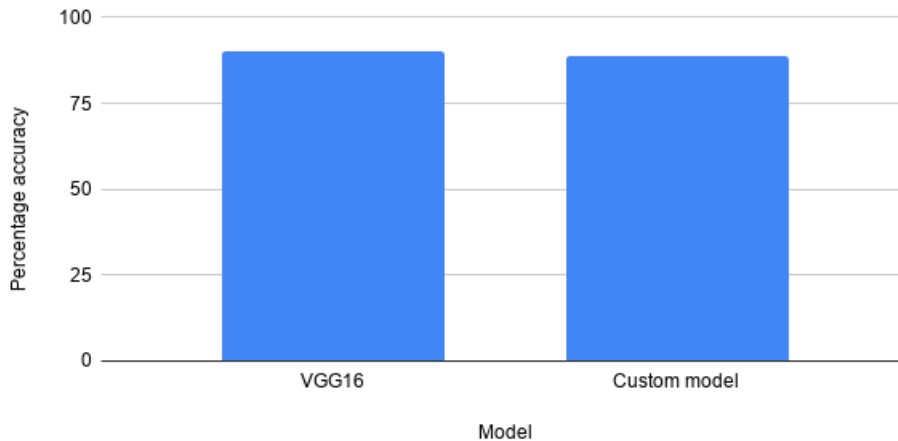


Figure 6.6: Percentage accuracy in classifying malware files from non-malware files vs. Model

It is evident from the results that when integrated with the blockchain the VGG-16 model, with a total of 4096 neurons, has a higher accuracy than the custom model, which has 128 neurons, in identifying malwares of different families as well as an overall higher accuracy when detecting malware and non-malware files correctly. However, the VGG-16 model consumes a significantly higher amount of time and computational resources when doing so which makes it more expensive. Further-

more, from our testing we have seen that the VGG-16 model lacked in stability in comparison to the custom CNN model.

6.2 Blockchain Output

When the user uploaded file is detected to be a malware, the blockchain function is terminated and a malware warning is shown.

```
Server response: ▶ {Output: "Malware"} bundle.js:2
Malware detected bundle.js:2
```

Figure 6.7: Detection of Malware

When the user uploaded file is not classified as a malware, the blockchain function is triggered and a block is mined.

```
Server response: ▶ {Output: "Not Malware"} bundle.js:2
Starting of the miner... bundle.js:2
senderAddress: Address1 receiverAddress: Address2 bundle.js:2
Block mined: bundle.js:2
000005a542e783f5cdad12a3e1f5830ed8fae60c80f01bc2952ba71a62957bca
Block successfully mined! bundle.js:2
Passing data to PHP... bundle.js:2
```

Figure 6.8: No malware detected, Blockchain is triggered

6.3 Limitations

We have faced numerous limitations while conducting our research. First and foremost, due to current circumstances, we did not have access to the Thesis Lab of BRAC University which would have provided us with the higher computational power required for the implementation of our thesis. For the testing and training process of our CNN models, we had to implement and run sophisticated machine learning algorithms using very limited resources. We especially struggled during the VGG-16 model training and testing process because it required maximum CPU usage, with each epoch taking quite a lot of time to finish. Moreover, after a certain number of epochs we faced kernel and system crashes. This limited us in the sense that we could not test the CNN models under different research conditions, for example, increasing the epoch, steps per epoch etc.

Chapter 7

Conclusion and future work

7.1 Conclusion

The once claimed unbreakable blockchain technology is now seen to become vulnerable against the ever growing cyber threats. Therefore, our main motive was to come up with an architecture where an efficient malware detecting neural network integrated with a blockchain based model could detect such attacks and stop them from infecting the said blockchain network. In this day and age where more and more people and organisations are adopting blockchain technologies for mainstream usages as well as the continually rising demands of cryptocurrencies, greater number of attackers are exploiting the security loopholes in blockchain based models. The idea of a combined architecture of a blockchain and a convolutional neural network to provide protection against malicious attacks is relatively new. Our model is a novel approach and only an initialization to a new walk of research. We were successful in designing and implementing a blockchain model which incorporates CNN in its architecture in such a way that any malicious data transactions cannot be made. This helps to eliminate any risk of hackers trying to bypass the blocks within the blockchain.

We have also compared the VGG-16 architecture with a customized one in order to find out which CNN model could detect malwares and prevent attacks in a more resource efficient way when combined with the blockchain. Our experimentation has shown that when integrated with the blockchain architecture, the VGG-16 model achieves a higher accuracy in comparison to the custom model. However, the VGG-16 model which has a greater number of layers is much more resource exhaustive. Therefore, in real world scenarios it would be wise for organisations to factor in these concerns when deploying such a model. We believe that the addition of convolutional neural networks to blockchains would be useful for organisations to add a layer of protection against malicious entites. Thus, we can say that with further improvements to our architecture it has a potential to be developed in full-scale for usage in real world systems.

7.2 Future work

Even though, our concept model shows potential in defending blockchain against malware attacks, there is certainly a lot of scope for improvement. One key challenge we faced was to figure out a cost efficient way to keep the stored files within the blocks without making them heavy. This can also help to eliminate the need of a centralized database. Another room for improvement can be implementing the CNN within the peer nodes. This can help to turn the private blockchain into a public blockchain, which eliminates any single point of authority. Furthermore, we tested our architecture with only two CNN models. Therefore, there lies a chance of better results if the various kinds of CNN models available are tested as well. Also, these models can be further optimized and trained with much larger data sets, especially VGG-16. Methods to allow faster response from server when loading heavier CNN models could be extremely useful. Moreover, a method to dynamically train the CNN models with new malware dataset collected from the malicious user inputs could be developed. Added to that, the blockchain can be deployed in a public server and analyzed in real time. Also, multiple encryption techniques other than SHA256 can be used to create hash and nonce. Therefore, testing in a public domain can prove essential to fully understand the potential of the model.

Bibliography

- [1] C. H. Kim, E. K. Kabanga, and S.-J. Kang, “Classifying malware using convolutional gated neural network. in: International conference on advances in computing and technology.,” Feb. 2018. DOI: 10.1109/CSE/EUC.2019.00095.
- [2] S. Morgan. (2020 (accessed: January 6, 2021)). “Cybercrime to cost the world \$10.5 trillion annually by 2025,” [Online]. Available: <https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/>.
- [3] A. Bera. (2019 (accessed: January 6, 2021)). “22 shocking ransomware statistics for cybersecurity in 2021,” [Online]. Available: <https://safeatlast.co/blog/ransomware-statistics/>.
- [4] M. Singh, G. S. Aujla, A. Singh, N. Kumar, and S. Garg, “Deep learning based blockchain framework for secure software defined industrial networks,” 2020. DOI: 10.1109/TII.2020.2968946,.
- [5] M. Saad, J. Spaulding, D. Nyang, L. Njilla, C. Kamhoua, S. Shetty, and A. Mohaisen, “Exploring the attack surface of blockchain: A systematic overview,” Apr. 2019. DOI: 10.1002/9781119519621.ch3.
- [6] B. Bambrough. (2019 (accessed: January 6, 2021)). “Warning issued after malware is found to have hijacked bitcoin blockchain,” [Online]. Available: <https://www.forbes.com/sites/billybambrough/2019/09/07/serious-malware-warning-over-bitcoin-blockchain/#4ce21be07c28>.
- [7] J. FRANKENFIELD. (2020 (accessed: January 6, 2021)). “Double-spending,” [Online]. Available: <https://www.investopedia.com/terms/d/doublespending.asp#:~:text=Double%5C%2Dspending%5C%20is%5C%20the%5C%20risk,power%5C%20necessary%5C%20to%5C%20manipulate%5C%20it>.
- [8] Vasa. (2019(accessed: January 6, 2021)). “How to use blockchains for spreading viruses? a study on use of distributed systems in malware deployment,” [Online]. Available: <https://medium.com/towardsblockchain/how-to-use-blockchains-for-spreading-viruses-690a5a4c65cf>.
- [9] Wikipedia. ((accessed: January 6, 2021)). “Convolutional neural network,” [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network?fbclid=IwAR1w1lKozT-9a2cM6SP2Z0fLNgY1wbOxxuTJuzthgtFvPBFL-5Zu_wpkSo0.
- [10] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, “Malware classification with deep convolutional neural networks.,” 2018.
- [11] A. Bakhshinejad and A. Hamzeh, “Parallel-cnn network for malware detection,” 2019. DOI: 10.1049/iet-ifs.2019.0159.

- [12] Z. Cui, L. Du, P. Wang, X. Cai, and W. Zhang, “Malicious code detection based on cnns and multi-objective algorithms.,” Mar. 2019. DOI: 10.1016/j.jpdc.2019.03.010.
- [13] M. MURRAY. (June 15, 2018 (accessed: January 6, 2021)). “A reuters visual guide: Blockchain explained,” [Online]. Available: <https://graphics.reuters.com/TECHNOLOGY-BLOCKCHAIN/010070MF1E7/index.html>.
- [14] J. J. Xu, “Are blockchains immune to all malicious attacks?,” 2016. DOI: 10.1186/s40854-016-0046-5.
- [15] O. Ajayi, M. Cherian, and T. Saadawi, “Secured cyber-attack signatures distribution using blockchain technology,” 2019.
- [16] J. GU, B. SUN, X. DU, J. WANG, Y. ZHUANG, and Z. WANG, “Consortium blockchain-based malware detection in mobile devices,” Feb. 2018. DOI: 10.1109/ACCESS.2018.2805783.
- [17] J. Ali, A. S. Khalid, E. Yafi, S. Musa, and W. Ahmed, “Towards a secure behavior modeling for iot networks using blockchain,” Jan. 2020. DOI: arXiv: 2001.01841v1.
- [18] A. Goel, A. Agarwal, M. Vatsa, R. Singh, and N. Ratha, “Deeppring: Protecting deep neural network with blockchain,” Apr. 2019. [Online]. Available: <https://www.researchgate.net/publication/33262989>.
- [19] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: Visualization and automatic classification,” Jul. 2011. DOI: 10.1145/2016904.2016908.
- [20] Microsoft. (2015 (accessed: January 6, 2021)). “Microsoft malware classification challenge (big 2015) classify malware into families based on file content and characteristics,” [Online]. Available: <https://www.kaggle.com/c/malware-classification/data>.
- [21] M. Mishra. (2020 (accessed: January 6, 2021)). “Convolutional neural networks, explained,” [Online]. Available: <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939#:~:text=Convolutional%5C%20Neural%5C%20Network%5C%20Architecture,and%5C%20a%5C%20fully%5C%20connected%5C%20layer>.
- [22] A. Zhang, Z. C. Lipton, L. Mu, and A. J. Smola. ((accessed: January 6, 2021)). “Padding and stride,” [Online]. Available: http://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html#stride.
- [23] —, ((accessed: January 6, 2021)). “Pooling,” [Online]. Available: http://d2l.ai/chapter_convolutional-neural-networks/pooling.html.
- [24] Arunava. (2018 (accessed: January 6, 2021)). “Convolutional neural network: An introduction to convolutional neural networks,” [Online]. Available: <https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05#:~:text=Fully%5C%20Connected%5C%20Layer%5C%20is%5C%20simply,into%5C%20the%5C%20fully%5C%20connected%5C%20layer>.
- [25] Prabhu. (2018(accessed: January 6, 2021)). “Understanding of convolutional neural network (cnn) — deep learning,” [Online]. Available: <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>.

- [26] C. Maklin. (2019 (accessed: January 6, 2021)). “Dropout neural network layer in keras explained,” [Online]. Available: <https://towardsdatascience.com/machine-learning-part-20-dropout-keras-layers-explained-8c9f6dc4c9ab#:~:text=Dropout%5C%20is%5C%20a%5C%20technique%5C%20used,update%5C%20of%5C%20the%5C%20training%5C%20phase.>
- [27] J. Jeong. (2019 (accessed: January 6, 2021)). “The most intuitive and easiest guide for convolutional neural network,” [Online]. Available: [https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480.](https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480)
- [28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [29] J. Brownlee. (2020 (accessed: January 6, 2021)). “A gentle introduction to the rectified linear unit (relu),” [Online]. Available: <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/#:~:text=The%5C%20rectified%5C%20linear%5C%20activation%5C%20function,otherwise%5C%20it%5C%20will%5C%20output%5C%20zero.&text=The%5C%20rectified%5C%20linear%5C%20activation%5C%20function%5C%20overcomes%5C%20the%5C%20vanishing%5C%20gradient%5C%20problem,learn%5C%20faster%5C%20and%5C%20perform%5C%20better.>
- [30] S. Sharma. (2017(accessed: January 6, 2021)). “Activation functions in neural networks,” [Online]. Available: [https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.](https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6)
- [31] M. Glossary. (2017(accessed: January 6, 2021)). “Loss functions,” [Online]. Available: [https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html.](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html)
- [32] K. Mahendru. (2019(accessed: January 6, 2021)). “A detailed guide to 7 loss functions for machine learning algorithms with python code,” [Online]. Available: [https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/.](https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/)
- [33] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” Apr. 2015. DOI: arXiv:1409.1556.
- [34] R. Thakur. (2019 (accessed: January 6, 2021)). “Step by step vgg16 implementation in keras for beginners,” [Online]. Available: <https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c#:~:text=VGG16%5C%20is%5C%20a%5C%20convolution%5C%20neural,competition%5C%20in%5C%202014.&text=It%5C%20follows%5C%20this%5C%20arrangement%5C%20of,by%5C%20a%5C%20softmax%5C%20for%5C%20output.>
- [35] L. Conway. (2020 (accessed: January 6, 2021)). “Blockchain explained,” [Online]. Available: [https://www.investopedia.com/terms/b/blockchain.asp.](https://www.investopedia.com/terms/b/blockchain.asp)
- [36] T. K. Sharma. (2019 (accessed: January 6, 2021)). “Public vs private blockchain: A comprehensive comparison,” [Online]. Available: [https://www.blockchain-council.org/blockchain/public-vs-private-blockchain-a-comprehensive-comparison/.](https://www.blockchain-council.org/blockchain/public-vs-private-blockchain-a-comprehensive-comparison/)

- [37] IBM. ((accessed: January 6, 2021)). “What is blockchain technology?” [Online]. Available: <https://www.ibm.com/blockchain/what-is-blockchain>.
- [38] TecraCoin. (2019 (accessed: January 6, 2021)). “What is genesis block and why genesis block is needed?” [Online]. Available: <https://tecracoin.medium.com/what-is-genesis-block-and-why-genesis-block-is-needed-1b37d4b75e43>.
- [39] M. Vidrih. (2018 (accessed: January 6, 2021)). “What is a block in the blockchain?” [Online]. Available: [https://medium.com/datadriveninvestor/what-is-a-block-in-the-blockchain-c7a420270373#:~:text=The%5C%20blockchain%5C%20is%5C%20a%5C%20chain, his%5C%20body%5C%20\(block%5C%20body\)](https://medium.com/datadriveninvestor/what-is-a-block-in-the-blockchain-c7a420270373#:~:text=The%5C%20blockchain%5C%20is%5C%20a%5C%20chain, his%5C%20body%5C%20(block%5C%20body)).
- [40] Y. Takefuji and H. Szu. (2019(accessed: January 6, 2021)). “Blockchain is vulnerable against classic database approach,” [Online]. Available: <http://medcraveonline.com/MOJABB/blockchain-is-vulnerable-against-classic-database-approach.html>.
- [41] V. L. Lemieux, “Trusting records: Is blockchain technology the answer?,” 2016.
- [42] J. FRANKENFIELD. (June 27, 2020 (accessed: January 6, 2021)). “Nonce,” [Online]. Available: <https://www.investopedia.com/terms/n/nonce.asp#:~:text=A%5C%20nonce%5C%20is%5C%20an%5C%20abbreviation, blockchain%5C%20miners%5C%20are%5C%20solving%5C%20for>.
- [43] P. Kotamraju. (May, 2019 (accessed: January 6, 2021)). “What is a nonce in block chain?” [Online]. Available: <https://www.tutorialspoint.com/what-is-a-nonce-in-block-chain>.