**Dipak Chandra Mistry**

**Id-20049246**

**Topic: Malware Analysis using Recurrent Neural Network**

**Declaration of Authorship**

I, Dipak Chandra Mistry, declare that this thesis titled, *"Malware Analysis using Recurrent Neural Network"* and the work presented in it are my own. I conﬁrm that:

• Where I have consulted the published work of others, this is always clearly attributed or referenced.

• Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

• I have acknowledged all main sources of help.

Signed: Dipak Chandra Mistry

Date: 24/12/2021

**Copyright 2021 by Dipak Chandra Mistry**

All rights are reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission from the author.

# ACKNOWLEDGMENT

Developing this research work initially seemed highly challenging due to requirements for conducting research in the field of deep learning, but it has been made possible by the support of some people to whom I would like to express my sincere gratitude.

First and foremost, I would like to thank my supervisor, Prof. . He provided me with a perfect balance of guidance and freedom. He became my guide in the world of deep learning – first, planting the seed of curiosity in the area and then encouraging my research. Thank you for helping me to have carried out research and for all your valuable advice in developing the document. He teaches me the standards and guidelines for scientific research and writing. Thanks to Professor, I managed to organize the process from the earliest stage and comply with deadlines.

I would also like to express my special thanks to my friends who inspired me to study deep learning. They triggered my interest in artificial neural networks that simulate a real brain. Our talks about brain activity further strengthened my decision to do this research.

Last but not least, truly heartfelt and great thanks to my family for the fantastic support and always being on my side, giving me the strength to continue and conclude this work.

Sincere thanks to all,

Dipak Chandra Mistry

# ABSTRACT

By using machine learning there are different Conventional techniques for detecting attacks that heavily rely on hand-crafted attributes that completely depend on subject specialists' expertise. Final learning strategies use the actual bytecode for inputs as well as attempt to extract some collection of informative attributes. Since the latter may perform adversely in situations when there is not enough information or even the dataset is unbalanced. The Recurrent Neural Network (RNN) is presented in this work to solve the challenge of malware identification and tracking by mixing multiple sorts of information to uncover the connection between various categories. Our method works on different malware families as a classification model. This method is an extensive solution of malware classification on the Microsoft malware dataset. In the end, the method achieves comparable results with machine learning algorithms.

**Keywords:** Classification, Machine Learning, Malware analysis, RNN

Table of Contents

[ACKNOWLEDGMENT 3](#_Toc91590976)

[ABSTRACT 4](#_Toc91590977)

[LIST OF FIGUERS 6](#_Toc91590978)

[Background 7](#_Toc91590979)

[Introduction 8](#_Toc91590980)

[Rationale 9](#_Toc91590981)

[Research questions 10](#_Toc91590982)

[Significance 11](#_Toc91590983)

[Limitation 12](#_Toc91590984)

[Dissertation structure 12](#_Toc91590985)

[Literature review 14](#_Toc91590986)

[Static analysis 19](#_Toc91590987)

[Dynamic analysis 20](#_Toc91590988)

[Stages of the malware analysis 22](#_Toc91590989)

[Methodology: 24](#_Toc91590990)

[Dataset 25](#_Toc91590991)

[Recurrent Neural Network 27](#_Toc91590992)

[Implementation 29](#_Toc91590993)

[Result and Discussion 43](#_Toc91590994)

[Evaluation Matrix 45](#_Toc91590995)

[Conclusion 51](#_Toc91590996)

[Reference list 52](#_Toc91590997)

# LIST OF FIGUERS

[Figure 1: Methodology work flow 24](#_Toc91590710)

[Figure 2:Work Flow of RNN and Feed Forward Neural Network 28](#_Toc91590711)

[Figure 3: Activation function 34](#_Toc91590712)

[Figure 4: RNN Model Summary 38](#_Toc91590713)

[Figure 5: Model Performance Graph 45](#_Toc91590714)

[Figure 6: Confusion Matrix of malware analysis 49](#_Toc91590715)

# Background

It is worth mentioning that a proposal is being developed based on the analysis

of Malware and work isolated environment by utilizing machine learning techniques. By developing this particular proposal it will be possible to create a better understanding of work isolated environments and Malware. It is important to describe that Malware is considered as intrusive software which is mainly designed to destroy and damage computer systems and computers. The common examples of Malware involve ransomware, adware, spyware, Trojan viruses, worms, and viruses. Most commonly businesses concentrate on preventive tools to stop the breaches. It has been identified that several advanced Malware can make their path into a particular network (Yan *et al.*, 2019). As an outcome, it is very necessary to deploy efficient technologies that will continuously detect and monitor malware that has avoided perimeter defences. Advanced malware protection needs high-level network intelligence and visibility along with different layers of safeguards.

Malware is also known as any file or program that is very harmful to a server or network and a computer. It is required to mention that Malware can easily infect devices and networks and is created to harm those networks and devices. All types of malware are mainly created to exploit the devices for the profit of hackers who have designed the Malware. In the first stage, this proposal will properly focus on analyzing dynamic and static analysis in malware and specifically focus on Trojan Malware and its features (Aslan and Samet, 2020). In the second stage, the proposal will describe how machine learning can be utilized with Python to perform the analysis of Malware. In addition to these, this proposal will also provide adequate information about the types of Malwares specifically Trojan Malware and it will also help to create a clear understanding of how Trojan Malware can be figured out.

# Introduction

In recent years, the amount and intensity of cyberattacks have increased to the point where cyberthreats have become considered is one of most pressing worries for the coming years. The significance of cyberwarfare on our daily lives should never be overlooked; researchers have recently seen it disrupt crucial elections and damage firms overnight. In 2015 and 2016, hackers infiltrated the Democratic National Committee (DNC) computer system and published confidential material in a collection that contained roughly 19,0 0 0 emails and 80 0 0 attachments from the DNC. According to Symantec (Chandrasekar et al., 2018), the number of new ransomware assaults discovered in 2016 tripled, and virus infections climbed by 36%. The global malware market is estimated to be worth millions, if not billions, of dollars and is rapidly expanding. Criminals, gangs, and even nation states are buying harmful software, cyber-capabilities, and products at a rapid rate in the underground service market. It has grown into a strong ecosystem, designed to take advantage of every opportunity and flaw in an increasingly interconnected world. For example, malware developers attempted to mine bitcoins this year by stealing customers' computing resources or by obtaining their cryptocurrency wallet information (Cleary et al., 2018; Daniely et al., 2018). It is vital to upgrade computer systems' cyber defences in order to stay up with virus growth and to be able to mitigate the impact of cyberattacks. Endpoint protection is an important part of any defence strategy.

Malware detection techniques are divided into two groupings: static analysis-based identification and dynamic analysis-based diagnosis. Static analysis, on the one side, examines a program's structure without running it. Dynamic analysis, on the other hand, monitors the behaviour of the programmed in the system. Experts then manually create a set of criteria to detect present and incoming risks depending on the info acquired from both static and dynamic analysis. The number of malicious threats was quite modest decades ago, and basic hand-crafted criteria were frequently sufficient to detect threats. However, due to the tremendous rise of malware streams in recent years, anti-malware solutions can no longer rely simply on costly hand-crafted rules. As a result of its ability to generalise in regard to never-seen-before malware, machine learning has become an intriguing signature-less approach for identifying and classifying malware. Conventional machine learning techniques rely on feature extraction to extract a set of discriminant features that offer a feature representation of malware that a classifier can use to assess if an executable is dangerous. To overcome the constraints of end-to-end training, we introduce the malware classification work with Recurrent Neural Network (RNN). Where when input pass to the neuron and neuron remembers their input, and pass it to the next neuron. After complete training of model, we achieve a strong classifier model. And we compare our achieve results with the state-of-the-art method like SVM, SVN, Random Forest model and Gradient Boosting models.

## Rationale

This rationale section will properly describe the reason behind developing this particular proposal. To analyze Malware and work isolated environments by utilizing python machine learning, this particular proposal is being developed. By adequately developing this proposal, it will be possible to create a greater knowledge about the isolated work environment and the types of Malwares. The proposal is being developed to create an in-depth notion about the dynamic and static analysis in malware. To gather adequate information and create a clear interpretation of Trojan Malware and its features, the researcher is developing this particular proposal (Or-Meir *et al.*, 2019). By developing this proposal, it will also be possible to know about the process or tools through which Trojan Malware can be figured out. It is very necessary to describe that as the entire social life switched to the digital world, developing countries can experience some risks. So this proposal can help to inform the developing countries about the risks which developing countries can experience due to the shift of social lives into the digital world. So these are the main reasons behind developing this particular proposal. It can be stated that this proposal will provide detailed information about work isolated environments and Malware by utilizing machine learning python.

## Research questions

Some research questions are as follows,

* What is the Malware work isolated environment using machine learning algorithm?
* How can Neural Networks be utilized with Python to perform the analysis of Malware?
* How much accuracy will be achieved by utilizing Neural Networks for the classification of Malware types?

**Aim**

The aims of this particular proposal are as follows,

* To analyze work isolated environments and Malware by appropriately utilizing machine learning.
* To inform developing countries about some risks which they can experience due to the shift of social life into the digital world.
* To focus on analyzing dynamic and static analysis in malware by using the tools.
* To create a clear interpretation of Trojan Malware and its features and to know how it can be figured out.

**Objective**

The objectives of this proposal are as follows,

* By using python machine learning, to properly analyze work isolated environments and Malware.
* Another objective of this proposal is to create an artifact for the same purpose.
* To install windows10 and Python and its IDE in any Operating system.
* Properly fix the Python to isolate the environment.
* To install, it would require software such as PyCharm and some libraries to building Malware analyze.
* To identify the process of specifying the Malware within the network.
* To identify the way of containing as well as measuring its damages.
* To find out what the suspect binary can do.

## Significance

The main significance of this particular research is to create a clear interpretation of the efficacy of the current model and how it would be beneficial as well as effective in identifying the issues. It is important to research because the entire social life is switching to the digital world and this digital transformation has created some opportunities and opened the window for hackers and cyber attackers to enhance cybercrime within the internet society(Ucci *et al.*, 2019). From this particular context, it is very important to conduct research regarding the Malware and work isolated environment by which people can get better solutions to protect their computer devices from cyber attackers. As technology is developing day by day and new technologies are introduced every day, the opportunities for cyber attackers to hack a computer device have also increased. If this continues for a long time, it can be very challenging to protect necessary information and personal data in computer devices (Shalaginov *et al.*, 2018). From this particular point of view, it is very necessary to research how the malware can be controlled by which a balance can be maintained with technological innovation. One of the main significance of this research is to get a better solution that can control the data breaching problem.

## Limitation

It is required to describe that the research is being conducted regarding the analysis of work isolated environment and the Malware by utilizing machine learning python. It has been identified that the limitations of the research are the characteristics of a methodology for a design that influenced or impacted the understanding of findings from the particular research. Research limitations are considered as the constraints that are placed on the capability to generalize from the outcome. The main limitation of this particular research is that it is not possible to find proper articles on this particular study topic, specifically those types of articles not possible to find which are linked to the analysis of work isolated environments and the Malware by using machine learning python.

## Dissertation structure

The dissertation structure will describe the process through which the research is being conducted appropriately. The overall dissertation structure is mainly separated into several chapters which include the introduction chapter, literature review, methodology chapter, result and discussion chapter, data analysis chapter, and recommendation and conclusion chapter. The introduction chapter will provide adequate information about the work isolated environment and Malware.

**Chapter 1: Introduction:** The introduction chapter will also describe the rationale of this research and also discuss the aim and objective of this research. The introduction chapter will develop some research questions by which the research can be conducted properly. It has been observed that the main aim of this research is to analyze work isolated environments and malware by using machine learning python addition, the introduction chapter will also explain the significance of this research. One of the main significance of this research is to create a clear interpretation of the efficacy of the current model and how it would be beneficial as well as effective in identifying the issues.

**Chapter 2: Literature review:** The literature review chapter will provide an overview of Malware analysis and will describe different types of Malware analysis. In the literature review chapter, the researcher will select different articles to collect proper information about the Malware analysis and work isolated environments by using machine learning python. The literature review chapter will help to create a better understanding of the study topic.

**Chapter 3: Methodology:** The methodology chapter will describe some components which will be utilized by the researcher to research efficiently. The methodology chapter mainly discusses the appropriate data collection method, research approach, research design, data analysis method, and research philosophy that need to be selected by the researcher to continue the overall research. This particular chapter will also describe ethical standards which need to be maintained by the researcher while conducting research. The methodology chapter will also discuss the reliability and validity of the research.

**Chapter 4: result and discussion:** The next chapter is the result and discussion which will provide necessary information which will be gathered by using Python. In the data analysis chapter, the researcher will compare the solution with the information which is collected from the literature review.

**Chapter 5: Conclusion:** The next chapter is the conclusion and recommendation in which the researcher has to summarise all the data points and data findings by which a clear interpretation of the study topic can easily be created. After that, the researcher will develop some recommendations.

# Literature review

Machine learning techniques toward malware/spyware identification and tracking can indeed be classified into several categories. The first is the static technique and the second technique is the dynamic technique. Static technique by (D., Semenov, et al., 2016; Siyi, Yuxin, Z., 2017) and the other dynamic is by (Tajoddin, et al., 2017; M., Sami, et al., 2015; Salehi et al., 2017).

The Static or rigid techniques require feature extracted devoid requiring malware operation, but the second dynamic technique necessitate the system's activation. Due to its capacity to find hidden malware and spyware by recognizing the changes extracted by historic information, machine learning algorithms seem particularly interesting for identifying and classifying dangerous malware programs. Acquiring existing data, cleaning up data and getting ready data, constructing algorithms, evaluating, and installing in operation are all part of the machine learning pipeline.  conventional machine learning techniques require cleaning the application, extraction of features, filtering, and minimization as part of the data planning method. All other attributes are then utilized to build a classifier to tackle this issue in mind, whether this is detecting malware and spyware or classifying malware and spyware onto groups. The Feature selection technique is often used in classic machine learning approaches that can derive discriminating characteristics from such a computer algorithm which offers the abstraction perspective that such a classification algorithm takes to build judgments about just the information. The latter part is training techniques, but it is on the other hand, combining features selection techniques, thereby substituting the previously outlined features design way with such a completely coachable machine. In this paper, we give the latest assessment of machine learning algorithms addressed to that same challenge of malware and spyware recognition and characterization (Hosseini and R.,2018; Aniello, et al., 2019).

Here is a list among the most significant static or rigid techniques available in the literature or composition, organized by kind of inserted information.

N-gram research was used to create that the very earliest algorithms that were completely based on machine learning applications. A continuous series of n elements from the textual content is the N-gram. Based mostly on the origin of information, the elements within this area could be bytes quantities or machine code directions. Working with big n-grams, but on the other hand, is operationally very expensive since the variety of different possibilities gets bigger and bigger with N. Experts offered several ways for learning N-gram-like signature of different particular patterns sans needing to enumeration all N-grams throughout the learning process. To categorize Microsoft and Smartphone malware, (Planes et al. 2017 and Bejar et al. 2017) developed a deep neural networks design that can be able to retrieve N-gram similar type of signatures and particular patterns from a series of operation codes mean opcode (Brandon et al. 2018 and Bálek et al. 2018) created final algorithms that learned characteristics again from the binary notation of compiled code through layering 1 or maybe more CNN architecture.

Malware and spyware developers frequently use encrypting as well as packaging techniques to disguise any harmful material and shield it from decrypting & discovery. Because encryption, as well as packaged chunks of coding, must have more unpredictability than endemic machine code, unpredictability measurement had already traditionally been being used to identify their existence. (J. and Ham 2007), for example, studied a sample containing data that also included simple text documents, local, compression, as well as encryption compiled code and concluded that the mean complexity was 4.347, 5.09, 6.80, and 7.17, correspondingly. Malware and spyware authors, on the other hand, were using a variety of strategies to get beyond conventional volatility detectors. As nothing more than a consequence, academics began looking at what has been termed run-able intrinsic volatility (Ivan and Sorokin, 2011). it seems to be, an application being classified into more non-overlapped sections that are either a specific distance, the volatility of every piece is measured. it is nothing more than some result, every item can be expressed in form of just a data series of volatility (Zhao and Chisholm. 2016) created a way for calculating a degree toward which differences inside a document's intrinsic volatility constitute them, suspect. In addition, (Planes et al. 2018b) suggested a malware and spyware classification scheme fully dependent on a convolutional network.

The run-able file type has a lot of intriguing characteristics as an input. Removable Compiled code files, for instance, hold data about the accompanying intellectually stimulating modules, program portions, including their relative proportions, along with many other things. Implementation of Software Development Kit (SDK) and API methods including framework operations, in particular, gives detailed data us about different operating systems that includes Windows operating system, Android operating system, and Linux operating system facilities and benefits, that may also be utilized for the simulate program behaviour (Yousra, Wenliang Du, 2013; Yadegari et al., 2010).

Furthermore, in addition to creating a rather alternative reliable prediction algorithm, it's indeed customary to incorporate data regarding API constructs with some extra aspects of an individual (D., Semenov, et al., 2016; Chan et al., 2017; Yang et al., 2016).

Such service operations can also be represented like a regular grid, referred even Function Call Graph, which includes the node representing the procedures a computer algorithm (Joris, et al., 2011) understands and even the nodes representing the stored procedures. Joris and suggested another unsupervised learning method by utilizing the clustering technique, clustering method contains his power in the world of artificial intelligence (AI) to identify identical features. he suggested the DBSCAN clustering method that is the Density-Based Spatial Clustering of Programmed with Noise technique to identify malware and spyware based on structural connections across method call sequences. (Chan et al., 2017) introduced another supervised machine learning technique that is linear-time mechanism towards malware and spyware analysis. (The supervised term is used when we know all targeted labels in advance) called network vectors approximation and demonstrated how to properly integrate graph characteristics alongside non-graph feature values.

The reorganization of concludable machine code like a grey level matrix (Nataraj et al., 2011), where each byte (byte is constructed with the combination of eight-bit) is read while single pixels inside the Two-Dimensional matrices and their values vary between Zero to Two Fifty-Five, is indeed a novel approach to express it (0 represents the black color pixel, while the next 255 represent the white color pixel value). It is indeed able to derive characteristics that describe the same styles inside a picture from any of this depiction, including such (GIST Jacob et al., 2011, Haralick D., Semenov, et al., 2016, Local Ternary Trends D., Semenov, et al., 2016) but also Principal component analysis (Narayanan et al., 2016) comes equipped that even a classification algorithm is using to categories malware and spyware. Furthermore, (Planes et al., 2018c and Kumar et al., 2018) investigated any use of CNN as the state of the art machine learning technique convolution neural network models that based on mathematical operations to identify the existence of particular characteristics and relationships in the picture data and matrix data that would be used to classify malware and spyware.

Furthermore, using only one type of feature to improve decision-making and categorize malware and spyware in an existing work environment is inadequate, as malware authors' concealment efforts may hide one or maybe more supervised machine learning characteristics. As a result, these attempts are indeed undertaken to create methods that really can interpret a variety of properties and gives appropriate results. Based on how the characteristics are merged, an acceptable practice could well be broken down into two categories. Introductory or cooperative sensing techniques, at however one side, entail combining numerous origins of information into separate vector representation which is then fed in such machine learning methods. (D., Semenov et al. 2016), for example, proposed a classification solution that incorporates various feature different kinds (electron density fact sheet, represents a movement, regularity of machine code, tally sheets, signifiers, as well as Desktop Application Frameworks) together in an individual functional unit that can be worn to learn boosting methods. Later or ensemble learning techniques, but on the other hand, combine the judgments of several classifications learned in different techniques. (Chan et al., 2017) developed an ensemble of independent malware analysers that important considerations malware and spyware, using a deep neural network to analyze binaries material portrayed as that of a picture as well as a fully convolutional network feeding using bytecode N-gram attributes as feed to demonstrate the concept. However, as far we know the literature, we did not get any approach that utilizes Recurrent Neural Network for analyzing the malware and spyware from their different families. In this work, we are going to implement the Recurrent Neural Network which will classify the types of malwares and spyware.

## Static analysis

It is very necessary to describe that there are different types of Malware analysis in which static analysis is one of them. According to (Pan et al. 2020), static analysis helps to check a malware file or program without running this file or the program. The author also described that static malware analysis is considered as one of the safest processes to analyze malware appropriately because executing the particular code can infect the system. It has been identified that within its most common form, static analysis gathers data from the Malware without perceiving the code. Hashes or MD5 checksums can be contrasted with a database to define whether the Malware has previously been recognized or not. The author also explained that advanced static analysis is also called code analysis. Some technical indicators are found out like hashes, file names, and strings like file header, domains, and IP addresses data could be utilized to define if the file or the program is malicious or not.

Tools such as network analysis and disassemblers can be utilized to recognize the malware without running it to gather data on how this malware functions. As stated by (Damodaran et al., 2017), sophisticated malware can involve malicious runtime behaviour which can remain unchecked since static analysis does not properly run the code. For instance, if a file or a program creates a string that mainly downloads a malicious program or file based on a dynamic string, it can remain unchecked by a common static analysis. The author also explained that static code analysis can provide insights into the errors of code. It has been identified that there are various types of static analysis tools, some of them are specific to particular kinds of errors, specific languages, and security. It has been observed that most static analysis tools are driven by specific rules, so it is very necessary to ensure that these rules are aligned with the objectives of the organization. It is worth mentioning that static code analysis saves money and time. The amount of time relies heavily on the number of tools utilized.

Static code analysers are always looking for different patterns that are determined as rules which could be the cause of code problems and security vulnerability. As described by (Shatnawi et al., 2019), static analysis can be utilized within a project where it can efficiently analyze the code without the execution of the code. The author also discussed that static analysis can scan the whole code and can also identify susceptibilities evening they are present within the particular corner of an application. It has been identified that static analysis can allow an individual to identify the defects in the earlier stage and it can also help to reduce the overall price which is required to fix them. These authors also describe that static analysis also helps to develop code security. In the current situation, as everything can be run on software, it is very necessary to properly evaluate the code for probable vulnerabilities from various points of view.

## Dynamic analysis

The dynamic malware analysis is a code that is put into the user system and this code enables one to keep an eye on the system to detect the infected section of the system. The dynamic analysis creates a safer environment around the system and it is also known as a sandbox. (Afianian et al., 2019) argued that the malware by the usage of the internet can access the user computer system as well as one able to remotely control the computer system. Many times, depending on dynamic malware analysis they can get all confidential data. Many governments by collaborating with the cyber security expert organization build software that enables them to prevent the malware. The dynamic malware analysis code has been accessed in the cloud computing system and helps them to keep the confidential data safe. According to (Carlin, O’Kane, and Sezer 2017), since the computing system or internet system was invented, many engineers were working on the antivirus to avoid the malware threat but in today's modern era the malware is much advanced so the antivirus was unable to prevent the malware attack. Many antivirus companies have been researching to identify a better way to prevent malware. Also, this antivirus helped the organization in its dynamic analysis. Also, dynamic analysis enables them to analyze different types of malwares that are Ramnit, Lollipop, Kelihos\_ver3, Vundo, Simda, Traceur, Kelihos\_ver1, Obfuscator.ACY, Gatak.

Using dynamic malware analysis creates several challenges for the user. The user has used different types of software to avoid malware activity in their computer system but many times they observe while installing other companies' new update software, the old software company shows their advertisement in their computing system. (Jeon, Park, and Jeong 2020) stated that in today's modern era people mostly depend on artificial intelligence (AI) along with the Ai technology that has been enabling them to keep their valuable data safe from malware. Most of the AI technology companies have been constantly providing an update of the Ai technologies because time by time the malware has been updated. Also, AI technology helps them to do a dynamic analysis of malware. In the Ai technology-based software program, if they notice any malware then they immediately inform the authority along with creating a firewall to prevent the malware activity. (Darabian et al., 2020) observe that most cyber attackers are financial motivators and they mostly target those systems that have stored financial data. So tech financial companies have been constantly updating their malware prevention software. Nowadays most economic transactions took place on the internet and maximum people have been depending on cryptocurrency for their financial transactions. So those organizations that maintain this type of financial server likely used the dynamic malware financial analysis system. The dynamic analysis enabled them to deeply analyse every binary code. Also, the dynamic analysis method is the most simple and easy as they prefer this malware analysis method.

## Stages of the malware analysis

**Static property analysis**

The static property analysis enables the user to observe closely the malware activity in the computer system. During analysis, the analysis at the time observes a suspicious file and examines the static property of the file. Static properties include a rope of joint into a file, hashes, packet signature, header details, and embedded resources along with metadata such as the creation data. The statistical analysis process enables an analyst to examine a specific static file or data. According to (Sihag, Vardhan, and Singh 2021), the android platform has been designed by several multiple security mechanisms and that mechanism helps them to prevent malware activity. Each layer of the android platform has its security mechanism. The analyst closely observed each layer's static part and if they found anything suspicious, then took the specimen for further analysis. The analyst used a lot of software to examine the static property of each file. Depending on the static property analysis the user avails to keep their valuable data safe.

**Interactive behaviour analysis**

Every malware file has a separate behaviour to run. After the analyst collects the static property they send it to the investigation laboratory. Where they analyze the source of the static proper and how this file runs along with observing the behaviour of the malicious file program. It has been proved their flexibility to understand each property of malware file has been helping them to create an effective firewall for preventing malware activity (Hampton, Baig, and Zeadally 2018) stated that malware software is defined as software that is the effect of the general operation of the operating system or this software may steal data of the operating system. In the last decades depending on the malware behavioural analysis, many updates in the operating system have been made that have made the operating system much stronger to prevent malware. The authority of the specific cooperating system has been constantly examining the suspicious file. If anything, they observe is wrong in their behaviour then they provide an update on an emergency basis.

**Fully automated analysis**

The easiest way to access the suspicious file is to use fully automated tools for better outcomes. This fully automated tool enables analysts to analyse the whole suspicious file within a short time. This tool mostly depends on AI technology. While the specific automated tool scans the suspicious file, if they observe anything dangerous for the system then they can shut down the illegal activity as well as many times this tool creates an effective firewall. (Ahmed, Kocher, and Al-rimy 2020) argued that Ransomware is the most sophisticated malware and the analyst is unable to detect the malware manually, so on a large scale, the analyst depends on automated malware detection tools. The malicious activity of the ransomware took place by tricking the user to download the payload. If the user downloads this kind of payload, then the ransomware can access all data of the user. During this time the automated tools warn the user and create a burden for the user to download this kind of payload.

# Methodology:

In this Malware analysis project, I am utilizing the power of machine learning. Machine learning (ML) is a powerful instrument for harnessing artificial intelligence (AI) technology. For its decision-making and cognitive capabilities. Machine learning repeatedly alludes to AI; however, it is a subfield of Artificial intelligence. This was a component of AI's progression till the late 1970s. Then it split out and began to evolve on its own. Machine learning has emerged as a critical response mechanism in cloud technology, eCommerce, as well as several other snipping technologies.

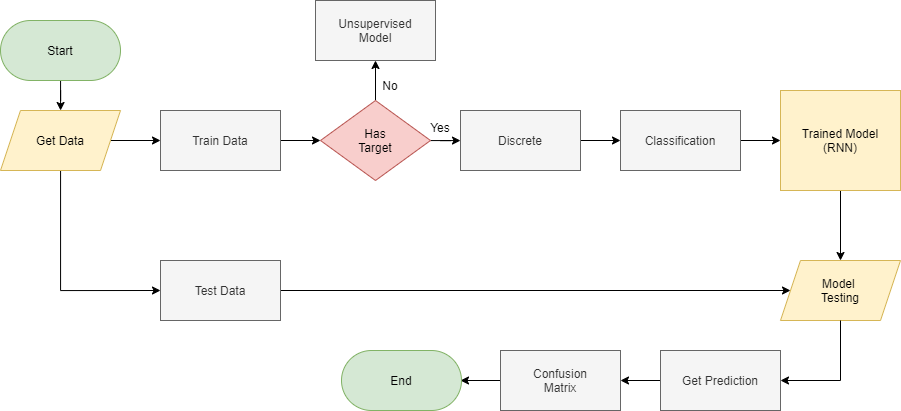


Figure 1: Methodology work flow

The above figure shows the general work flow of methodology, first we get data and ensure that our data contains target label or not, if it contains target label it means the data fall in supervised learning category on the other hand if it does not contain than we have to deal it as a un-supervised learning. In our malware dataset it shows towards supervised learning, after this we check our data belongs to discrete label it means we have to perform classifications. In next step we just train our RNN model and test model on testing dataset, and evaluate results.

## Dataset

In the malware analysis work, we utilize the Microsoft malware dataset from Kaggle for dealing with classification problems (Kaggle Challenge by Microsoft 2015). Kaggle is a Google LLC unit, it is an international forum of data analysts and machine learning experts. Users can use Kaggle to locate and release data sources, with a digital machine learning platform, investigate and construct models, collaborate with other data scientists and engineers who specialize in machine learning, and participate in contests to tackle data science problems and win prizes. This available dataset contains training and testing features separately in the form of csv (comma separated file). The training csv file has its own feature value as and, on the other hand testing csv file has its own features. The training csv has 1805 columns (these are our features that used for training the model) and it totally consist on 10,868 records that represent 9 different families of Microsoft malware. Every malware item seems to have an Id and class, which is a 20-character hash value that identifies it. And an integer that represents one of nine possible malware family names:

Ramnit Lollipop Kelihos\_ver3 Vundo

Simda Tracur Kelihos\_ver1 Obfuscator.ACY

Gatak

**Ramnit:** Through linking to a remote computer, it creates a loophole.

A remote hacker can use this loophole to do a variety of tasks on the affected PC, such as download files, remove files and launch files.

**Lollipop:** It's commonly packed with the Player Plus software and installs even without the user’s permission. It sends information about the device's geolocation and security software to a remote server.

**Kelihos\_ver3:**  This is another malware that belongs to Microsoft malware family. Any malevolent attacker might use this malware to gain authority and control of the computer. This virus is transmitted through unsolicited emails that contain links to additional infections. any malware has power to install it.  Computers can indeed interact with many other Computers to exchange intelligence about spamming user inbox, stealing users’ confidential details, or downloading and running dangerous software.

**Vundo:** Vundo belongs to a virus series with many components which display "from out contexts" catchy advertising. Additional programs, such as spyware or malware, could well be downloaded & operated by members of the series. Vundo indeed that codename of something like the danger connected also with Vundo virus family's is command and control.

**Simda:** Simda also belongs to a collection of rootkits that can thieve personal data that includes account number, account name, account credentials, even certifications. This uses recordkeeping & Web injections to steal data. This also even runs hidden operations and putting compromised computers' lives in jeopardy.

**Traceur:** Traceur is another type of malware and belongs to a malware group that may reroute user internet that affects search results. Hackers utilize this to make a fraud like getting money for something like the virus creators by defrauding internet ad networks. Such Trojan horses take over many usual and unusual sites and redirect them to their websites.

**Kelihos\_ver1:** Kelihos\_ver1 seems to be a family of trojans that sends out spam emails and text messages. The spam communications might find references to Kelihos\_ver1 spyware launchers. The virus may interface with distant sites to share data that would be used to carry out a variety of operations, such as distributing unsolicited emails, collecting confidential material, and obtaining and running files directly.

**Gatak:** Gatak malware also belongs to a malware family. The majority of gatak is in it to make money off of you. The crooks describe several harmful programs designed to thieve your account information like credit details as well as card information, net payment credentials, and also many other personal details for illicit purposes.

**Obfuscation:** obfuscation Spyware has also belonged to the type of malware. obfuscation seems to be a technique for making written as well as digital material is harder to decipher. It assists attackers in concealing crucial keywords (also referred to as threads) that a software employs since they indicate trends in infection behaviour. Register entries but also corrupted Webpages are instances among these values.

## Recurrent Neural Network

In this malware analysis project, we have utilized the state-of-the-art Neural Network that is a Recurrent Neural Network (Hochreiter Sepp et al., 1997). The RNN algorithm is majorly used when we have data in the form of a sequence. This was the first algorithm in the world of AI that has internal storage and keeps data safe. As a result, it's ideal for neural network models involving sequential information. Recurrent neural networks are quite a sort of neural network which is both powerful and reliable. Because that's the only one with internal storage, it is one of the most intriguing methods currently in use. Recurrent neural networks, like so many other deep learning approaches, are relatively new. It first was developed in the early 1980s, however, it has only been in the last few years that we have realized its real greatness. They can remember things due to internal storage that’s why it easily predicts. That's because they're the chosen approach for time-series data, voice, texts, financial information, music, multimedia, temperature, and many other types of sequential data.

On the other hand, there is another state-of-the-art neural network that is Feed-Forward Neural Network (Bebis, and Georgiopoulos 1994), the flow of information goes in only one direction in Feed Forward Neural Network means it directly passes towards hidden layers and then moves to the final output layer. Feedforward Neural Network has no memory space where it stores any information that’s why it doesn’t save the previous information of neurons. And this is also one main reason why feed-forward neural network predicts very well. And in the same way, Recurrent Neural Network stores the passing information in their respective neuron’s memory cell. The input is cycled through such a spiral in RNN. Whenever it produces a judgment, it takes into account the input signal as well as what it has learned from prior inputs. The below figure shows how the recurrent neural network works differently from the feed-forward neural network.

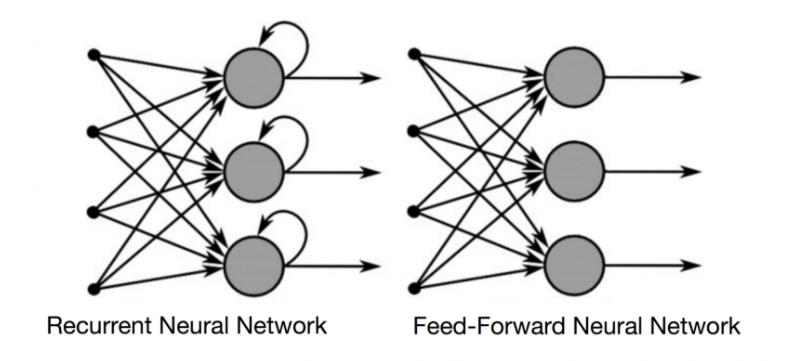


Figure 2:Work Flow of RNN and Feed Forward Neural Network

## Implementation

In the implementation phase, we also utilize TensorFlow, which is an open-source framework that we used for building a neural network. It enables programmers to construct new neural network models with various layers. Classification, observation, grasping, finding, forecasting, and synthesis are some of the most common uses for TensorFlow. Before building a recurrent neural network, we have to perform some pre-processing on the given dataset. Data preparation refers to the steps involved in transforming or encoding data so that it could be readily understood by a computer. In the real world, the majority of the dataset contains data heterogeneously and our data mining algorithms did not perform very well on it. This means using data mining techniques upon the skewed data will yield bad outcomes. It could be unable to recognize trends properly. So, data pre-processing is important and plays a vital role in increasing the accuracy of data mining techniques.

Data pre-processing consists of four different types:

* Data Integration
* Data Cleaning
* Data Transformation
* Dimension Reduction

**Data Integration:** Aggregation or integration of data is a majorly used and dominant step in the data pre-processing stage. integration of data indeed the feature extraction approach who brings together information by contacting a diverse provider as well as presents it to consumers that looks something like cohesive manner.

Numerous datasets, the matrix is given, and record systems are examples of those same providers. Creating a company's current centre is amongst the prominent examples of data integration steps. Data integration is required, particularly when attempting to address a true problem such as recognizing a threat and Ct Images revealing the existence of tumors.

**Data Cleaning:**  data cleaning is a very important step in machine learning, before jumping towards the implementation of any machine learning algorithm we have to perform a variety of steps for the evaluation of our dataset, and validate that our data seems to lave or not, actually in effect at the learning stage of ml model if the data is not cleaned and it directly passes to any machine learning deep learning model, in this scenario the model also learns that raw or irrelevant features and after this case how we aspect our machine learning and deep learning model predicts well. so, the data cleaning process includes locating as well as removing mistakes in what seems like a piece of information that could harm a prediction system. Data cleaning encompasses a broad range of actions and practices aimed at detecting and correcting difficulties. It is a step in the data pre-processing process that requires taking in null values, normalizing errors, resolving inconsistencies, and removing noise.

**Data Transformation:** Performing transformations of data is a data preparation approach that reorganizes and restructures original details in that way it seems enough to big data may quickly and readily uncover valuable intelligence. sometimes transformation of indeed crucial while transferring information into a database online data centre. transformation of data simpler for the examination phase also it seeks of trends whenever given information looks uniform or quite fine. Transformation of the data plays a vital role in the data pre-processing method which should be applied to information regarding the mining of data in this way for producing upcoming trends it seems to be simpler to comprehend. Assuming that you have a basic understanding of transformations, you can proceed onto another collected material approach that has been explored. We must next aggregate the qualitative data once the material has indeed been cleared. by altering its content, shape, or formatting of material through the use of techniques like Normalization, Generalization, etc

Data Transformation completely consists of six different methods:

* Smoothing
* Aggregation
* Generalization
* Discretization
* Normalization
* Construction of Attributes

**Smoothing:** In the smoothing phase of the transformation of data we simply remove the noise factor from the available dataset. smoothing approach is being cast-off to reduce the maximum amount of distortion that is located inside in our available data. The corrupted or nonsensical information inside a dataset thought to those as cacophony. The smoothing makes the cast of techniques emphasize the information's unique properties. when the noise from available data has been removed, the procedure may identify any little alterations to find unusual trends. in this way, the approach will detect all changes in the information and any trends.

**Aggregation:** In this step of the transformation of data we have to perform an aggregation operation on the available data. Performing aggregate procedure on the available big quantity of given information, data analysis reduces the size of collected data.

For instance, we are given a collection of purchase history from a company that seems to be monthly earnings throughout the year. there is an option we may combine complete information and generate the company's sales volume record.

**Generalization:** The generalization step in the transformation of data includes conversion of data between two different types, in this phase we have to perform some replacement like we replace the data that belongs to low level into high-level data, but we are utilizing the concept of hierarchy chain.

**Discretization:** The discretization step in the transformation of data mainly includes converting uninterrupted consist sequence tiny gaps. In real-time, many data Gathering tasks need continuous properties. Most contemporary data gathering systems; however, are incapable of handling the properties.

**Normalization:** The normalization step in a transformation of data is a process of reducing the range of information quantities to either a narrower range, including (-1.0 to 1.0) or (0.0 to 1.0). there are several approaches used in the normalization step to applying the normalization on the given data the first approach of normalization is Min-max normalization that is based on data linearity. The one another normalization technique that is the second approach having name z score normalization. In this way, the value of an attribute is normalized based on its standard deviation. The last normalization technique is the decimal scaling method, which can perform normalization by simply altering the position of the available attributes to each other.

**Construction of Attributes:**

In attribute construction of transformation of data, the additional features are built by either using the available values while reading from the current collection of characteristics in that way establish collected information which facilitates feature extraction.

**Dimension Reduction:** The dataset in a database system could be too huge for data processing and data information extraction to handle. One option is to create a simplified version of the information that is substantially less in size but delivers comparable research findings.

In this malware analysis work we specifically used dimension reduction, we have almost 1805 features in the dataset. For reducing the dimension, we just load our training dataset and split all the features and target labels from a dataset. After splitting the training features, we initialize our Principal Component Analysis (Narayanan et al., 2016) method that performs dimensionality reduction of a dataset. After performing dimensions reduction, we have the remaining 784 features for our dataset, now it's quite enough for the training purpose of neural network.

After applying dimension reduction successfully, we used one other term that is Data Transformation that basically normalizes our training dataset. It converts the integer numbers in the range of 0 to 1. This means here our data is standardized or normalized by using this our model also learns patterns of data very easily and also, we achieve good accuracy. In pre-processing, we also utilize an encoding method that is a label Encoder () from sklearn pre-processing library that can convert the string data in the form of an integer. It is important step to do. So that the machine learns from data easily.

**Network Architecture:**

After encoding label into categorical we start constructing our Recurrent Neural Network model, our model consists of a total of 5 layers. The first layer is input that takes input data and next, we have three hidden layers after the hidden layer at the end we have our last output layer of the model. Recurrent Neural Network layers interpretation is below:

**The input layer of the RNN model:** tf. Keras. layers. Input (28,28)

**Hidden layer of RNN model**

**1st hidden layer:** tf. Keras. layers. Time Distributed (tf. Keras. layers. Dense (32)

**2nd hidden layer:** tf. Keras. layers. SimpleRNN (128, return sequence=True)

**3rd hidden layer:** tf. Keras. layers. SimpleRNN (128, return sequence=False)

**The output layer of RNN model**: tf. Keras. layers. Dense (9, activation = ‘SoftMax’)

In the second layer we set the return sequence True the input and return to neuron and in the third layer we false the return sequence it means here our neuron stores the input as memory. And our last layer is the output layer that shows we have 9 output labels. And we use the activation function as SoftMax.

**Activation function:** is a function that is added into neural network layers to activate the neuron so that our network learns tricky pattern easily. As the term suggests, activation function activates a neuron by firing its output by calculating its weighted sum and adding bias with it. We use activation functions to introduce non-linearity into the output of a neuron. The simplest type of activation function is a threshold activation function. It just fires the one if weighted sum is greater than a threshold value otherwise zero as shown below.

Diagram

Description automatically generated with low confidence

Figure 3: Activation function

This type of activation function gives binary output at the neuron. But in real time problem, we cannot use this activation function as one object is similar to other with some probability. We can use linear activation function but for gradient decent the linear activation function will result in a constant value. When we will calculate error in pack propagation, no error will be calculated even if it occurs. So, there is no use of linear activation function too. Hence, non-linear activation functions are very suitable to use for real world data. Let’s discuss some of these.

These are the following activation function that we used in this work:

**ReLU**: stands for rectified linear unit, is an activation function that was introduced by Nair and Hinton in 2010. It is a piecewise non-linear function that outputs the input back if positive, otherwise, it outputs zero. The mathematical representation is:

f(x)=max (0, x)

It is a widely used activation function for deep learning problems for the reason it is easier to train and achieves better performance.it means Rectified linear activation function, If the input is true, it will be output directly; else, it will be produced as zero.

**Sigmoid**: At the beginning stages of deep learning, the Sigmoid function seems to an utmost commonly utilised activation function. This is an easy-to-deduce as well as smooth algorithm. The sigmoid function is S shape logistic function, the sigmoid function also has ranged from 0.0 to 1.0. A sigmoid function value generally distinguishable by itself and, you may simply determine overall inclination of sigmoid line everywhere at different locations. the activated value is the gain in the output neuron value and the remaining value is the loss rate.

**Tanh**: is a non-linear activation function that activates the with the limited range It is also known as the S-shaped activation function. The Tanh function is remarkably identical to sigmoid activation function and it is non-linear in nature, It is a sigmoid function in scaled factor, and even has the similar to sigmoid, it also has a resultant value in the range of [-1,1]. Tanh is another extensively being used effective activating mode.  The higher an input, it means the closest to result to 1.0, as well as the lower the inputs the closest the result to -1.0. Another thing to note seems to be that tanh has a greater slope versus sigmoid.

**SoftMax**: The SoftMax activation function applies is output layer of the neural network to activate each class vector separately. The SoftMax function scale the output probabilities. It is used in multinomial or multi classification problem, in our work we have to predict nine different families of malware that’s why the SoftMax is used here in the last layer of RNN model.

After completing the model architecture, we just need to compile the model before starting training. For model compilation we used model. Compile command.

In compilation time we use ‘Adam’ as an optimizer function and our loss is ‘categorical cross entropy’ and our metrics is [accuracy]

**Optimization techniques**: have been used in machine learning for change given model's attributes. The major goal of many optimizers seems to be to optimise a losses technique by adjusting model’s weight, and control learning rate and loss rate because in every neural network model training our goal is to reduce the cost/loss function. A losses method is actually use for to estimate how effectively the model works. In this sense there are variety of optimization techniques like RMSprop, SGD and Adam are available but, in our work, we utilized the Adam as optimization function.

**Adam:** Adams is indeed an efficient machine learning approach that calculates individualized education plans for various characteristics. This optimizer is computationally intensive, uses minimal space, and is well-suited to scenarios involving massive data as well as variables.  it combines Adapt Gradients as well as Root Mean Square Propagation, and two SGD (stochastic gradient descent) techniques.

**RMSprop:** For training any deep learning model, RMSprop seems to be a gradient-based strategy. Geoffrey introduced this backpropagation method. when our data is   via extremely complicated processes such as deep learning model, gradient tended both to evaporate or expand that shows gradient problem being solved.  It is created like a stochastic small learning algorithm.

It solves problem by just normalising gradient with such a floating sum of the square’s gradients.

RMSprop, essentially expressed treats the model learning rate like an adaptive factor rather than a hyper - parameters. It indicates also learning fluctuates with the pace of time.

**SGD:** The SGD (Stochastic Gradient Descent) is a simple and basic optimization function that takes the responsibility for converging the deep learning models. The stochastic word means, anything that have a system or specific procedure that could be link to an arbitrary probability. there are also many multiple gradient descent algorithms. in the Stochastic Gradient Descent scarcely any samples from dataset and pass for each iteration these samples are selected by random probability and pass forward, rather than we pass the whole dataset for the iteration. In Stochastic Gradient Descent there is another term that called batch, the batch plays also a big role in the SGD processes. it shows that how many numbers of samples are selected from the actual dataset for each iteration. the batch term is used for calculating the gradient value for each iteration.

The summary of or Recurrent Neural Network model is given below that shows all layers and trainable parameters, total parameters and non-trainable parameters. We get this summary simply using model. summary () function.

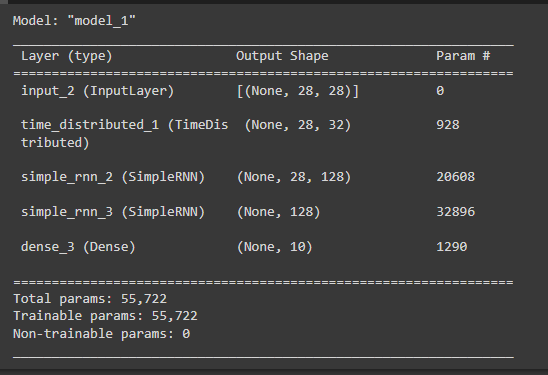


Figure 4: RNN Model Summary

This summary shows that we have total five layers in the model architecture, and our total numbers of parameters are 55,722, in which the trainable parameters are 55,722 and non-trainable parameters are 0. So, our model is train on all parameter because we are doing any transfer learning work. The transfer learning is the process in which we get the features of any pretrain model that are already trained on larger dataset, and we just load that pretrained model into our python script and remove their last dense layer. And after removing the dense layer we concatenate our new model layers and start training the model. The main benefits of transfer learning we achieve the good accuracy easily. But in our work after compiling the model and getting its summary, the next step is training our model on training data. On training time, we use the model. Fit command and pass training data for training purposes and validation data for validation purposes. We start train our model on 20 epochs and set the batch size value is 32.

**Epochs**: is the important parameter of fit command epochs mean how many time our whole dataset pass to the network for training purpose. Epoch is a hyperparameter defined for a network which means how many times our network will work through the provided dataset during training. Its value ranges from one to infinity. Greater the number of epochs, more the number of time network will see all the data. An epoch may of consists of one or more batch size. If the batch size is one, and the number of epochs is 100 and weights will be updated 100 times during an epoch. And if the batch size is two the weights will be updated 50 time.

**Batch size:** is also an important parameter that we use in the fit command it means how many samples of data pass in one step. When a model is on training mode, it continuously changes its parameters like weights, biases, and sometimes learning rate as there are numerous numbers of calculations performed at that time. We pass the data to model in different ways. We can either give complete data or provide one sample or even chunks of data at a time. This chunk of data is called batch size. More precisely, it is defined as a hyperparameter that defines to number of data samples to work with after which the network updates its other hyperparameters like weights etc.

Let’s talk about what batch sizes we can have. When we pass our data at once i.e., batch size is equal to the size of training data then it is called Batch Gradient Descent. We can also pass one example at a time; it is called Stochastic Gradient Descent. But the most popular way is we define a real number like 32, 64, 128 etc. less than size of training data and greater than one as our batch size. It is a popular way to assign batch size and is called Mini-Batch Gradient Descent.

**Batch Gradient Descent.** Batch Size = Size of Training Set

**Stochastic Gradient Descent.** Batch Size = 1

**Mini-Batch Gradient Descent.** 1 < Batch Size < Size of Training Set

We start with many epochs, often hundreds or thousands, allowing the model to change its parameters until minimize the error. You may have seen examples of epochs as 50, 100, 200, 500, 1000 and so on.

**Implementation Libraries**

To implement this malware analysis, I have used the Python programming language with PyCharm IDE. As I have experimented with different methods, all of the above-mentioned methods were implemented using the python programming language.

**Python**

Python has long been the programming language of choice for programmers in Machine Learning (ML) and Artificial Intelligence (AI). Python provides developers with several sophisticated tools, some of which assist improve their productivity while also reducing the burden of their job. Other than the fact that Python was explicitly created to support AI, ML, and DL. Here are some of the many characteristics that set Python apart from other programming languages:

**Libraries of Python**

I have used the following Python libraries for this project to implement all the methods discussed in the results sections.

* **Tensor flow**
* **Keras**
* **Matplotlib**
* **Numpy**
* **Pandas**
* **Scikit-Learn**

**Tensor flow:**

TensorFlow is a deep learning framework that is fast, versatile, and scalable, and it is used in both research and production. Working with Machine Learning on Python is made easier with TensorFlow, one of the best available libraries. For both beginners and professionals, Google's TensorFlow makes deep learning models development simple. We can take the help from this deep learning framework to build our models and retrain the models.

**Keras:**

Kernel-based neural network libraries for Python are very popular, and Keras is one of the most popular. After being developed by a Google developer for the open-source ONEIROS operating system, Keras was quickly included in the TensorFlow Core Library and made available to users who wanted to use it on top of the TensorFlow architecture. For constructing a neural network, many components and tools are provided in Keras, including the following: Cost functions and Activation, Neural layers, Objectives, Dropout, Batch normalization, and pooling.

**Matplotlib:**

Matplotlib is one of the most sophisticated data visualization tools in Python. Matplotlib is focused on making simple things simple and difficult things feasible. With a few lines of code, you can create plots like scatterplots, histograms, bar charts, and other different types of visualization.

**NumPy:**

Numpy is essential for computational science in Python. A variety of routines for rapid array operations, including random simulation, basic linear algebra, and many more other kinds of tasks, are available as part of a Python package called NumPy.

NumPy is based on the nd-array object at its heart. N-dimensional arrays of homogenous data types are encapsulated inside this. While many operations are converted into code for performance, they are done on arrays of homogeneous data types in N-dimensional form. In addition to those mentioned above, there are many key distinctions between NumPy arrays and Python's conventional sequence objects:

NumPy arrays are assigned a predefined size when created, while Python lists are not (which can grow dynamically). When you change an nd-array, a new array is created, and the old array is deleted. To have an element of the same data type in memory, all items in a NumPy array should have the same type of data. When doing arrays of different-sized items, you may use Python's arrays of objects (including NumPy).

The ability to do sophisticated mathematical and other operations on vast amounts of data in various fields is made easy using NumPy arrays. Actions like this often complete faster and use less code than Python's built-in sequences can handle.

**Pandas:**

Data manipulation and analysis are the primary functions of the Pandas Python data analysis package. In terms of training, it begins before the dataset has even been generated, which is advantageous. Pandas is a library that allows machine-learning programmers to deal with time series and organize multidimensional data quickly and easily. Pandas' most valuable capabilities are contouring and pivoting datasets, joining and merging datasets, managing missing data, and data alignment in terms of data management. These are just a few of the many ways datasets can be shaped rearranged. Pandas also have several other functional capabilities.

**Scikit-Learn:**

This python library has much been used in machine learning on numeric data. This product offers a simple way to integrate with several machine learning frameworks, including NumPy and Pandas. Scikit-learn offers many algorithms, including Clustering, Regression, Model Selection, Dimensionality, Reduction, Classification, and Pre-processing of the dataset.

Scikit-design's philosophy focused on data modeling, but it does not get involved with loading, manipulating, managing, and visualizing data. Researchers can utilize this library to develop end-to-end solutions.

# Result and Discussion

In this phase we discuss the results that we achieved after successfully training the Recurrent Neural Network Model on the above-discussed parameter we evaluate the model training by considering four major characteristics that are used for every neural network. The training accuracy, training loss, validation accuracy as well as validation loss.

**Training accuracy:**  Training accuracy is a parameter that plays a vital role for evaluating the model performance, it shows that how much our model learns accurately during training time. During model training phase we analyse its training accuracy that are shown at the start of every epoch. If the training accuracy move smoothly without any major ups and downs it means our neural network model training going fine.

**Training loss:** Training loss is an also another important parameter that we consider for the evaluation purpose of our model at training time. The loss values are the number of figures shows us how much our model loss the data during training time. The training loss value also available in every epoch, if the training loss value of last epoch is minimum like 0.01 it means our model is good on training time.

**Validation accuracy:** Validation accuracy is also a parameter that used for evaluate the model performance in the sense of there validation during training time. while training the model we used model.fit command and in this command, we also pass the validation data just like same as we pass the data for training. By validation accuracy parameter we notice our model validation accuracy at every epoch. If the validation accuracy is moves correspondingly with training accuracy it means our model learn very well during training time.

**Validation loss:** Validation loss is another important parameter that we consider during training time of our neural network model, the validation loss is same like the training loss, but these are the loss values that are calculated on the basis of validation dataset. The validation loss value is also shows at the end of every epoch.

When model training and validation is complete on training data as well as validation data respectively. We achieve training accuracy 97%, validation accuracy 98% and training loss of 0.07%, and validation loss is 0.04%.

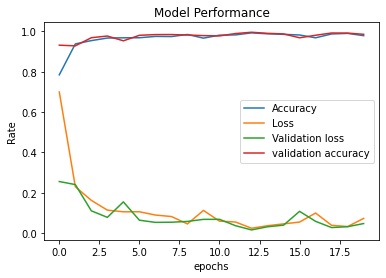


Figure 5: Model Performance Graph

The above graph shows model performance on both training and testing accuracy. In graph the x-axis represents the number of epochs and y-axis represents the rate values. and, blue line in the graph shows the training accuracy and yellow line shows the training loss of our model. On the other hand, red line shows the validation accuracy of our model and green line shows its validation loss value.

## Evaluation Matrix

We also evaluate our model performance by utilizing different evaluation metrics the first one is confusion matrix. And, F1-score, Recall, Precision and confusion matrix, etc.

**Confusion Matrix:** For a classification problem, we obtain some results that tell us how many examples are classified to which class after running a model. It is presented in a tabular form and the table is called confusion matrix. In short, a confusion matrix tells us a summary of prediction results of model or problem. In short, it is a way to infer how your classification model is confused when it is classifying each sample. For example, we provided our network with 1000 images of cats and dogs containing 500 samples for each class. The network classifies 450 and 455 instances of dogs and cats respectively correctly and misclassifies 50 and 45 instance of both. The confusion matrix will be as follows:

Table 1: Confusion Matrix Example

|  |  |  |
| --- | --- | --- |
| N = 1000 | **Actual**  **Dog** | **Actual**  **Cat** |
| **Predicted**  **Dog** | 450  (TP) | 50  (FP) | 500 |
| **Predicted**  **Cat** | 45  (FN) | 455  (TN) | 500 |
|  | 495 | 505 |

In the above table, we have classified each cell with some labels. Let us define these terms:

**True Positive or TP:** In these cases, we predicted yes (these are dogs). It means originally these were dogs and we predicted these as dogs.

**True Negative or TN:** We predicted cats or in other words we predicted no for dogs. It means these samples were originally cats and are predicted as it is.

**False Positive or FP:** We predicted a dog, but it was a cat. It is also known as a "Type I error."

**False Negative or FN:** We predicted cat, but they belong to dog class. It is also known as a "Type II error."

The advantage of confusion matrix is we can have a quantitative analysis of our model by calculating different metrices such as accuracy, precision, recall, F-score etc. Accuracy is defined as the ratio of correctly classified samples to the total number of samples as follows:

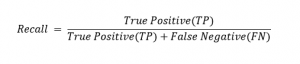
Text

Description automatically generated

Hence, accuracy in above given example according to above formula comes out to be 0.905.

**Recall**

It is a metric that is used to identify the degree of correct assumptions made by our model. The mathematical formula for recall is:



Recall gives us an idea of how correctly our model is identifying the similar data. It is also referred to it as Sensitivity or True Positive Rate. Hence, according to given formula od recall the above-mentioned example has recall of 0.909. It means our model 90% sure for correctly classifying a positive class.

**Precision:**

Now, we came to know about accuracy and recall. We know our model can identify 90% of positive classes but we need to know what portion of positive classification is always correct. In that case, the metric will help out is precision. It is defined as follows:

A picture containing text

Description automatically generated

The precision for above mentioned examples is 0.9. It means 90% of classified positive examples are correctly classified. It must be noted that the models with precision 1.0 do not have any False positive samples.

**F1 score:**

F1-score also referred to as F-score or F measure, is a metric that identifies equilibrium between precision and recall because sometimes increasing recall decreases precision and vice versa. F1 score is defined as:

Text

Description automatically generated with low confidence

It is represented as harmonic mean of precision and recall. According to give formula, F1 score for our example is 0.909. It is maximum when we have same precision and recall. As we can have trade-off between precision and recall, we can aim for maximum F1 score that would be thought as good evaluation metric. Even in some situations, like disease prediction, it is aimed to have a high recall value because we don’t want any suffering patient to be missed by our model. Whereas, in some other situations like fraud detection or loan defaulter bank would prefer the high precision that number of samples would be correctly classified. When we evaluate a model, we consider all these metrices to decide how good our model is performing.

We can generate all this confusion matrix and all related metrices using a library named sklearn. Sklearn also have much more metrices like ROC curve, precision recall curve etc. Just import related function from sklearn and calculate whatever you require. Following is a code example for sklearn.

|  |
| --- |
| from sklearn metrics import confusion matrix  from sklearn metrics import classification report  # Actual values from data  actual = [1, 0, 0, 1, 0, 0, 1, 0, 0, 1 ,0]  # Predicted values from data  predicted = [1, 0, 0, 1, 0, 0, 0, 1, 0, 0]  # Confusion matrix  Matrix1 = confusion matrix (actual, predicted, labels= [1, 0])  print ('Confusion matrix: \n', matrix1)  matrix2 = classification report (actual, predicted, labels= [1, 0])  print ('Classification report: \n', matrix2) |

The confusion matrix that we get from malware analysis is shown below:

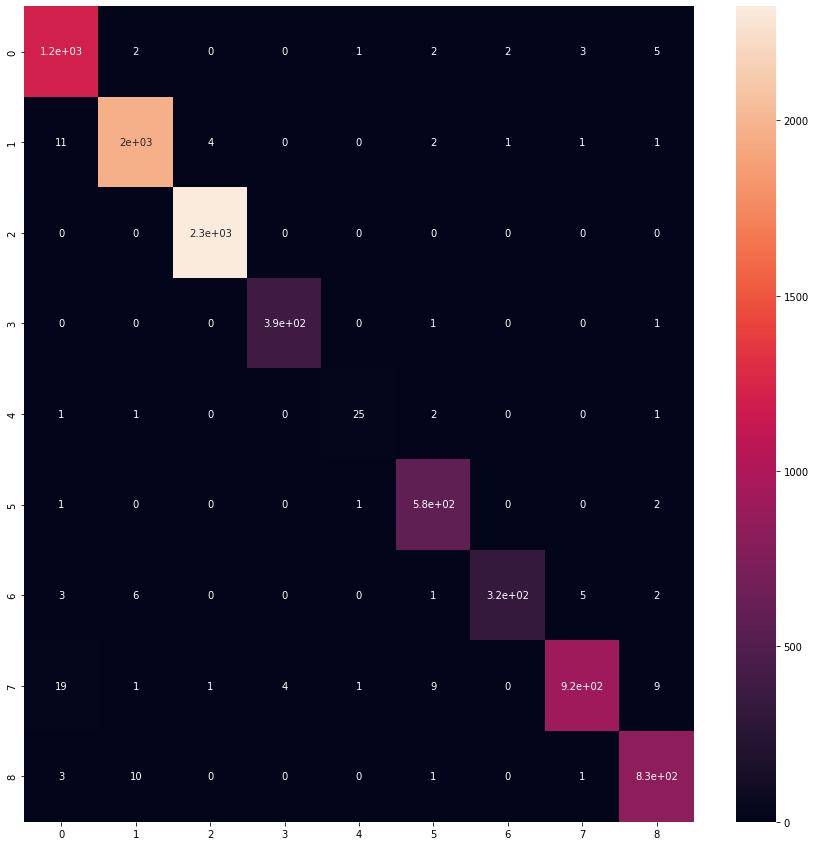


Figure 6: Confusion Matrix of malware analysis

Above figure shows the confusion matrix that generate by using the confusion matrix () function from sklearn library that show the impact of each class according to predictions.

After implementing all these evaluation metrics to our malware analysis, we achieve the following results that are shown in below classification report:

Table 2 : Classification Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classes** | **precision** | **recall** | **f1-score** | **support** |
| 1 | 0.97 | 0.99 | 0.98 | 1226 |
| 2 | 0.99 | 0.99 | 0.99 | 1989 |
| 3 | 1.00 | 1.00 | 1.00 | 2325 |
| 4 | 0.99 | 0.99 | 0.99 | 392 |
| 5 | 0.89 | 0.83 | 0.86 | 30 |
| 6 | 0.97 | 0.99 | 0.98 | 585 |
| 7 | 0.99 | 0.95 | 0.97 | 335 |
| 8 | 0.99 | 0.95 | 0.97 | 976 |
| 9 | 0.98 | 0.98 | 0.98 | 845 |
|  | | | | |
| **Accuracy** |  |  | **0.99** | **8694** |
| **Macro avg** | **0.97** | **0.97** | **0.97** | **8694** |
| **Weighted avg** | **0.99** | **0.99** | **0.99** | **8694** |

The above table shows the accuracy 99%, recall 99% as well as f1-score is 99%, this classification report is generated over the testing csv to our model and get prediction. For getting prediction we simply use the model. prediction () function. And pass the testing csv file as a parameter to this function and get results. After getting any prediction we use the classification\_report () function and pass these prediction as well as actual label to this classification\_report () function as parameters.

These are the results that we achieve from our Recurrent Neural Network. These results are compared able to the other researchers, on this time we are mentioning the results of (Gibert et al., 2020). They have implemented different machine learning algorithm from scikit (sklearn) library, that include Logistic Regression model, Support Vector Machine (SVM) as a linear kernel, Subversion(SVN) as RBF kernel, Random Forest model and Gradient Boosting model. The first Logistic Regression model gives maximum accuracy 97.94%, SVM gives 97.74, SVN gives 94.09, Random Forest gives 97.41% as well as last algorithm Gradient boosting gives 97.33% these are all the results that are mention by (Gibert et al., 2020). On the other hand our Recurrent Neural Network gives 99% accuracy which perform well on the microsoft malware validation dataset.

# Conclusion

In this research, we offer the finest and unique malware classification system that uses a customize architecture to incorporate characteristics that may belong to different malware families. To the best of our knowledge, the current study is one of the first studies that use Python machine learning, deep learning, and neural networks exclusively to perform classification on malware analysis data. So this achieved result is the better result of classification performance than the other classifiers that just accept one type of data as input. This framework may be enhanced by several additional feature kinds to describe the features of compiled code, and then it will be used even as a guideline for upcoming enhancements as well as other researchers.

The success of this strategy may be analysed and compared with other state-of-the-art machine learning procedures that are reported before. Our Recurrent Neural Network deep learning architecture's detection performance and f1-score are significant compared to those other attribute engineering-based data mining methods, to it, really depending solely on a single three major types of characteristics from Compact Executable code, and also much more precise than machine learning existing approaches.

# Reference list

Afianian, A., Niksefat, S., Sadeghiyan, B. and Baptiste, D., (2019). Malware dynamic analysis evasion techniques: A survey. *ACM Computing Surveys (CSUR)*, *52*(6), pp.1-28.

Ahmed, Y.A., Kocer, B. and Al-rimy, B.A.S., (2020). Automated Analysis Approach for the Detection of High Survivable Ransom ware. *KSII Transactions on Internet and Information Systems (TIIS)*, *14*(5), pp.2236-2257.

Aslan, Ö.A. and Samet, R., (2020). A comprehensive review on malware detection approaches. *IEEE Access*, *8*, pp.6249-6271.

Aniello, L., Baldoni, Ucci, D., R., 2019. Survey of machine learning techniques for malware analysis. Comput. Secur. 81, 123–147. doi: 10.1016/j.cose.2018.11.001.

Brandon, Raff, E., Barker, J., Sylvester, J., R., Catanzaro, B., Nicholas, C.K., 2018. Malware detection by eating a whole EXE. In: The Workshops of the The Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2–7, 2018., pp. 268–276.

Bálek , Krcál, ˇ M., Švec, O., M., Jašek, O., 2018. Deep convolutional malware classifiers can learn from raw executables and labels only.

Bebis, G. and Georgiopoulos, M., 1994. Feed-forward neural networks. IEEE Potentials, 13(4), pp.27-31.

Chandrasekar, K., Cleary, G., Cox, O., Lau, H., Nahorney, B., Gorman, B.O., O’Brien, D., Wallace, S., Wood, P., Wueest, C., 2018. Internet Security Threat Report. Technical Report. Symantec Corporation.

Daniely, Y., Clayton, R., Johnson, S., Eisner, T., Fishman, Y., Kaye, J., 2018. Security Report. Technical Report. Check Point Software Technologies Ltd.

Carlin, D., O’Kane, P. and Sezer, S., (2017). Dynamic analysis of malware using run-time opcodes. In *Data analytics and decision support for cybersecurity* (pp. 99-125). Springer, Cham.

Chan, Hassen, M. , P.K., 2017. Scalable function call graph-based malware classification. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. ACM, New York, NY, USA, pp. 239–248. doi:10.1145/ 3029806.3029824.

Damodaran, A., Di Troia, F., Visaggio, C.A., Austin, T.H. and Stamp, M., (2017). A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, *13*(1), pp.1-12.

Darabian, H., Homayounoot, S., Dehghantanha, A., Hashemi, S., Karimipour, H., Parizi, R.M. and Choo, K.K.R., (2020). Detecting crypto mining malware: a deep learning approach for static and dynamic analysis. *Journal of Grid Computing*, pp.1-11.

D., Semenov, Ahmadi, M., Ulyanov, S., Trofimov, M., Giacinto, G., 2016. Novel feature extraction, selection and fusion for effective malware family classification. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy. ACM, New York, NY, USA, pp. 183–194. doi:10.1145/2857705.2857713.

D., Siyi, Yuxin, Z., 2017. Malware detection based on deep learning algorithm. Neural Compute. Appl. doi:10.1007/s00521-017-3077-6.

Gibert, D., Mateu, C. and Planes, J., 2020. HYDRA: A multimodal deep learning framework for malware classification. Computers & Security, 95, p.101873.

Hampton, N., Baig, Z. and Zeadally, S., (2018). Ransom ware behavioural analysis on windows platforms. *Journal of information security and applications*, *40*, pp.44-51.

Hosseini, R., Souri, A., 2018. A state-of-the-art survey of malware detection approaches using data mining techniques. Hum. Centric Comput. Inf. Sci. 8 (1), 3. doi:10.1186/s13673-018-0125-x.

Hochreiter Sepp, Schmidhuber Jürgen Long short-term memory Neural Comput., 9 (8) (1997), pp. 1735-1780

Ivan, Sorokin, 2011. Comparing files using structural entropy. J. Comput. Virol. 7 (4), 259. doi:10.1007/s11416-011-0153-9.

Jeon, J., Park, J.H. and Jeong, Y.S., (2020). Dynamic analysis for IoT malware detection with convolution neural network model. *IEEE Access*, *8*, pp.96899-96911.

J., Ham, Lyda, R., 2007. Using entropy analysis to find encrypted and packed malware. IEEE Secur. Priv. 5 (2), 40–45. doi:10.1109/MSP.2007.48.

Joris, Kinable and Orestis Kostakis. "Malware classification based on call graph clustering." *Journal in computer virology* 7.4 (2011): 233-245.

Jacob, Nataraj, L., Karthikeyan, S., G., Manjunath, B.S., 2011. Malware images: Visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security. ACM, New York, NY, USA, pp. 4:1–4:7. doi:10.1145/2016904.2016908.

M., Sami, Ghiasi, , A., Salehi, Z., 2015. Dynamic VSA: a framework for malware detection based on register contents. Eng. Appl. Artif. Intell. 44, 111–122. doi:10.1016/ j. engappai.2015.05.008.

Microsoft (BIG 2015), Microsoft Malware Classification Challenge (BIG 2015), 2015, Accessed on: December 7, 2021. [Online]. Available: https://www.kaggle.com/c/malware-classification/overview

Or-Meir, O., Nissim, N., Elovici, Y. and Rokach, L., (2019). Dynamic malware analysis in the modern era—A state of the art survey. *ACM Computing Surveys (CSUR)*, *52*(5), pp.1-48.

Pan, Y., Ge, X., Fang, C. and Fan, Y., (2020). A systematic literature review of android malware detection using static analysis. *IEEE Access*, *8*, pp.116363-116379.

Planes, D., Bejar, J., Mateu, C., Planes, J., Gibert, , Solis, D., Vicens, R., 2017. Convolutional neural networks for classification of malware assembly code. In: International Conference of the Catalan Association for Artificial Intelligence, pp. 221–226. doi:10.3233/978-1-61499-806-8-221.

Planes, Gibert, D., Mateu, C., J., 2018. An end-to-end deep learning architecture for classification of malware’s binary content. In: Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4–7, 2018, Proceedings, Part III, pp. 383–391. doi:10.1007/978-3-030-01424-7\_38.

Shalaginov, A., Banin, S., Dehghantanha, A. and Franke, K., (2018). Machine learning aided static malware analysis: A survey and tutorial. In *Cyber threat intelligence* (pp. 7-45). Springer, Cham.

Shatnawi, A., Mili, H., Abdellatif, M., Guéhéneuc, Y.G., Moha, N., Hecht, G., Boussaidi, G.E. and Privat, J., (2019). Static code analysis of multilanguage software systems. *arXiv preprint arXiv:1906.00815*.

Sihag, V., Vardhan, M. and Singh, P., (2021). A survey of android application and malware hardening. *Computer Science Review*, *39*, p.100365.

Tajoddin, Bidoki, S.M., Jalili, S., A., 2017. PbMMD: a novel policy based multi-process malware detection. Eng. Appl. Artif. Intell. 60, 57–70. doi: 10.1016/j.engappai. 2016.12.008.

Ucci, D., Aniello, L. and Baldoni, R., (2019). Survey of machine learning techniques for malware analysis. *Computers & Security*, *81*, pp.123-147.

Yan, Y., Li, Z., Chen, Q.A., Wilson, C., Xu, T., Zhai, E., Li, Y. and Liu, Y., (2019, June). Understanding and detecting overlay-based android malware at market scales. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (pp. 168-179).

Yousra, Wenliang Du, Aafer, and Heng Yin. "Droidapiminer: Mining api-level features for robust malware detection in android." *International conference on security and privacy in communication systems*. Springer, Cham, 2013.

Yadegari, Sami, A. , B., Rahimi, H., Peiravian, N., Hashemi, S., Hamze, A., 2010. Malware detection based on mining API calls. In: Proceedings of the 2010 ACM Symposium on Applied Computing. ACM, New York, NY, USA, pp. 1020–1025. doi:10.1145/1774088.1774303.

Yang, Zhang, Y., Huang, Q., Ma, X., Z., Jiang, J., 2016. Using multi-features and ensemble learning method for imbalanced malware classification. In: 2016 IEEE Trustcom/BigDataSE/ISPA, pp. 965–973. doi:10.1109/TrustCom.2016.0163.

Zhao, Chisholm, Wojnowicz, M., G., Wolff, M., X., 2016. Wavelet decomposition of software entropy reveals symptoms of malicious code. J. Innov. Digit. Ecosyst. 3 (2), 130–140. doi: 10.1016/j.jides.2016.10.009.