Deep Learning in Malware Identification and Classification

Balram Yadav and Sanjiv Tokekar

Abstract Albeit the cyber world has become an essential part and the lifeline of the present day, there are threats associated with it. People access the cyber world for various services like networking, banking, communication, shopping, and for other uses. Malware is one of the primary and perilous threats among malevolent software for the decades in the cyber and the computing world. Due to its magnification in volume and in complexity, malware and its variant identification and classification are the most central and severe problems nowadays. Since malware inception, more and more malware is engendered and designed, as time passes; more intricate malware is designed enormously. Researchers and analysts are perpetually probing for a solution that is the most efficacious to fight back with malware. The most-famed methods utilized for malware analysis is signature-based detection, static, and dynamic analysis. In recent years, signature-based detection has been proven ineffective against the escalation of malware and its variants. Malware classification is attracting widespread interest due to its vast proliferation. In this chapter, we have chosen to discuss and explore another method of malware analysis that is image-based malware analysis utilizing deep learning. We are specifically discussing malware classification utilizing malware visualization and deep learning, one of the most widely implemented techniques in many real-world applications. To better understand the concept from a practical perspective, we additionally discussed and implemented a fundamental level malware classifier, for the reader's further research and study purpose. The main objective of this chapter is to avail readers a better and in-depth understanding of malware classification, visualization, deep learning algorithms and emerging challenges, open issues.

B. Yadav (✉)
I.E.T, D.A.V.V., Indore, India
e-mail: balram.dreamsworld@gmail.com

S. Tokekar
I.E.T., D.A.V.V., Indore, India
e-mail: stokekar@ietdavv.edu.in

M. Stamp et al. (eds.), *Malware Analysis Using Artificial Intelligence and Deep Learning*, https://doi.org/10.1007/978-3-030-62582-5_6

1 Malware and Malware Analysis

In this section, we are discussing what is malware, what is malware analysis, what is malware classification, how we visualize malware, etc. This section is prodigiously needed and avails readers to better understand the malware analysis and malware classification.

1.1 Malware

To efficaciously understand and analyze malware, you should be familiar with it. Malware is an abbreviation for malicious software (malicious software is an umbrella term used to refer a variety of forms of inimical software or programs) intending to access information, resources without the user's notification, and sanction. Malware is any code that performs inimical. Malware infections are among the most frequently encountered threats in the digital and computing world. Malware is additionally utilized for obtaining a password, obtaining confidential data; additionally, they are acclimated to trap the government. The malware is mall functioning software that is found on the computer systems. Malware and other threats are defined as specially indited programs to perform deleterious activities. An assailant designs malware to compromise computer services, access data, bypass access controls, and affects the functioning of a computer, its applications, or data.

The accelerated growth of devices in the cyber world has designated a massive obstruction in front of malware analysts, researchers, and additionally for antivirus companies. Assailants utilize the cyber world for illegitimate activities to commit financial frauds, to gain access to sensitive and personal information, to gain access for systems and networks. In recent years, there has been an expeditious increase in Internet attacks [7, 8]. The researchers and analysts customarily suggested security mechanisms and designed novel methods to fight malware and its variant attacks. There has been a great amendment in the design of malware. Afore the term malware was coined, all the malignant programs were considered under the term computer virus. Malware is an umbrella term for any program that contravenes the confidentiality, integrity, and availability of accommodations, contrivances, networks, or systems.

The list below provides an overview of variants of malware based on malware's behavior includes Trojans, viruses, worms, rootkits, botnets, phishing, spam, spyware, key loggers, logic bombs, etc.

Adware is kenned as advertisement software. Adware is the designation given to those programs which are designed to exhibit advertisements on your computer when you explore the cyber world, and then redirect your search requests to advertising websites and accumulate information about you and your interest. Adware is considered as malevolent because it amasses data without your consent or sanction. It is a type of malware that automatically distributes advertisements. Advertising-fortified software often comes bundled with software and applications and most of them serve as a revenue tool.

Virus A computer virus is a malevolent program that cyber attackers program to reproduce in massive amounts and affects the functioning of a computer and degrades its performance. It is also known as infectors. It conventionally does so by assailing and infecting subsisting files on the target system and from one host to another. Viruses must execute to do their deleterious task, so they target any type of file that the system can execute. A virus is a software program that modifies other programs and affixes itself to their code. A virus can run by itself; they perform intended malevolent activities when the infected program is executed.

Spyware Abbreviated for spy software (software that spies on a computer system). It is programmed to monitor and record browsing data as well as confidential information and other activities. It is a type of malware that spies and tracks utilizer activity without their erudition. The capabilities of spyware can include keystrokes accumulation, financial data harvesting, or activity monitoring.

Worm Functionally virus and worms are homogeneous. Worms are infectious and spreads. Assailers design worms to replicate themselves like a virus. However, a worm replicates without targeting and infecting specific files that are already present on a computer. They utilize a computer network to spread, relying on security failures on the target computer to access it, and steal or delete data. Worms are network viruses that can spread over the network by duplicating themselves. They do not transmute or ravage the user's files but they reside in main memory and duplicate themselves, and by this they make the system and network unresponsive.

Trojan A trojan or trojan horse is a maleficent program that represents its utilizer to be appearing utilizable and innocuous files or legitimate software. Attackers distribute trojans as routine software, game, or an implement that persuades a utilizer to install it on their computer. The denomination is derived from the antediluvian Greek story of the wooden horse that used to march into the city of Troy by stealth. Trojan horses are just as pernicious on computers and considered destructive. Cybersecurity experts consider trojans to be among the most hazardous types of malware, concretely trojans are designed to glom financial information from users.

Key logger A keystroke logger, or key logger, captures keystroke ingression made on a computer by the utilizer, often without the sanction or erudition of the utilizer. Key loggers have legitimate uses as a professional information technology monitoring tool . However, keystroke logging is commonly utilized for malefactor purposes, capturing sensitive information like usernames, passwords, answers to security questions, and financial information.

Rootkit A rootkit is a set of software tools, typically malevolent, which gives an unauthorized utilizer privileged access to a computer. Once a rootkit has been installed, the controller of the rootkit can remotely execute files and transmute system configurations on the host machine. Rootkits cannot self-propagate or replicate. They must be installed on a device.

Bots and Botnets Additionally kenned as robots. Bots are maleficent programs designed to infiltrate a computer and automatically respond to and carry out

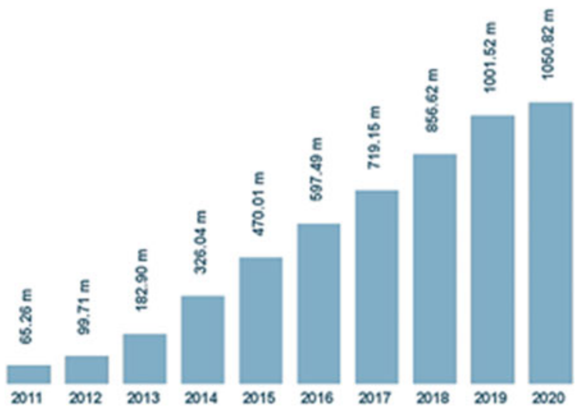


Fig. 1 Malware evolution statistics

instructions received from a central command and control server. Bots can self-replicate (like worms) or replicate via user action (like viruses and trojans).

Ransomware Ransomware is a type of malware that locks the data on a victim's computer, typically by encryption. The cybercriminal behind the malware demands payment afore decrypting the ransomed data and returning access to the victim. The motive for ransomware attacks is proximately always monetary, and unlike other types of attacks, the victim is conventionally notified that an exploit has occurred and is given instructions for making payment to have the data renovated to normal. It is a type of malignant software that essentially restricts utilizer access to the computer by encrypting the files or locking down the system while injunctively authorizing a ransom. Users are forced to pay the malware author to remove the restrictions and gain access to their computers.

1.1.1 Current Scenario of Malware Magnification

This section deals with the current scenario of the magnification of malware and its variants. We can see from Fig. 1 that the number of attacks is growing every year. The number of malware found perpetual to increment because malware and its variants can be engendered utilizing automated tools and reusing code modules. Reports from different antivirus companies limpidly describe that number of malware, and its variants are incrementing expeditiously.

A report from the av-test institute verbalized that in the period 2011- august 2020, 1050.82 million malware were recorded [7] and 10.87 million new malware were reported in the month August 2020 Fig. 2.

One more report from McAffe antivirus company placidly describes the statistics of malware evolution, millions of malware and variants are discovered [8].

There are many more reports from different antivirus companies conspicuous the fact that malware and its variant assailments are incrementing every year and besides malware, reports additionally present the current scenario of attacks of Internet of

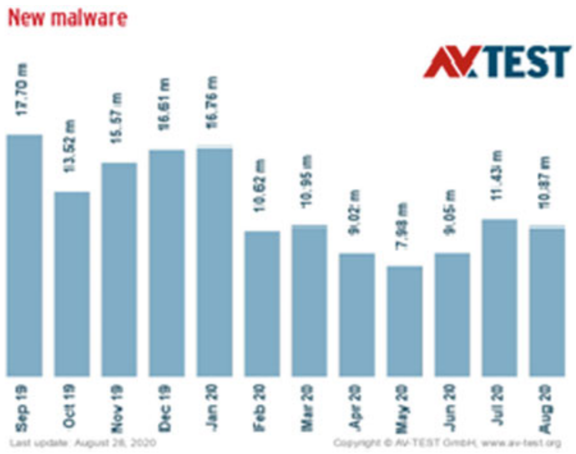


Fig. 2 New malware evolution statistics

Things (IoT) malware, mobile malware, and withal an expeditious increase is in ransomware recently. With these statistics, manual malware analysis is not feasible anymore; it does not scale to handle this enormous count of malware that's why the process of malware analysis needs to be automated. This is discovered or reported malware and its variants. It does not account for malware that has not been discovered or reported yet. There could be millions more out there that are still relishing the comfort of not being detected

1.1.2 Malware Family

A malware family is a group of malware that comports and functions in the same way. A family can be divided into different variants, especially if an incipient malware has different functionality and structure than the precedent ones. Malware family is the term utilized for the malware samples that belong to the same family designates they apportion their code or can have homogeneous code, capabilities, damage potential, inchoation, or behavior. Malware family betokens that incipient malware is designed by utilizing antecedent malware so we can group them in a single malware family.

For example, the Loylda family refer Table 1 of malware has four known variants: Loylda.AA1, Loylda.AA2, Loylda.AA3 and Loylda.AT, malware samples from malimg dataset [19].

1.1.3 Threats From Malware

The damage caused by malware depends upon it, whether it infected a computer, a business organization or whole network. The consequences of the damage caused by malware depend upon the type of malware. There are many threats associated with variants of malware, such as some malware interrupts the services of the system and operating system, some accesses file system without sanction, some access user's confidential data, some perform a denial of accommodation attacks, some minimize the space of a system, effacing, misplacement and corrupts files, some access systems resources, they additionally decelerate the process of the system, engender multiple shortcuts, automatically consumes an abundance of space in the system and truncating the recollection of the system. Malware greatly affects the functionality of computers and networks. Malware additionally causes hardware failure.

1.2 Malware Analysis

Malware analysis is the process of inspection and dissection of the functionality, purport, inception, and potential impact of malevolent code. In another way, it is the process of extracting cryptic information from malware code through static, dynamic, or hybrid inspection by utilizing tools, techniques, and methods. The data that is extracted from malware can be simple like its file type, strings to more perplexed information like malfeasance. Malware analysis denotes analyzing and inspecting binaries of malignant code to understand its working and finding methods for identification and classification of homogeneous files. Attributes or properties of data/samples, and these attributes are analyzed to engender paramount insights into the data under analysis. We accumulate features from malware binaries.

For example, in the facial detection system, the features would be shape, size, color, and structure of eye perceivers, nose perceiver and in malware analysis, features can be strings from the malware binaries, application programming interface (API) call sequences, n-grams, etc.

1.2.1 Traditional Approaches

The investigation of maleficent code is done traditionally mainly with static, dynamic, and hybrid analysis. Traditional approaches Fig. 3 such as static, dynamic, or hybrid analysis extract separate levels of features from malignant samples for identification and relegation, which cannot perform efficiently and accurately. The utilizations of deep learning for malware classification offers an expedient of building scalable machine learning models, which may handle any scale of data, without expending of resources such as memory. Deep learning marks malware depend on the general pattern, which directs the distinguishing of a variety of malware attacks and their variations. Furthermore, deep learning conducts a profound classification and improves its accuracy because deep learning identifies more features than conventional machine learning methods by passing through many calibers of feature extraction. This enables deep learning models to acquire an incipient pattern of malware after the fundamental training phase.

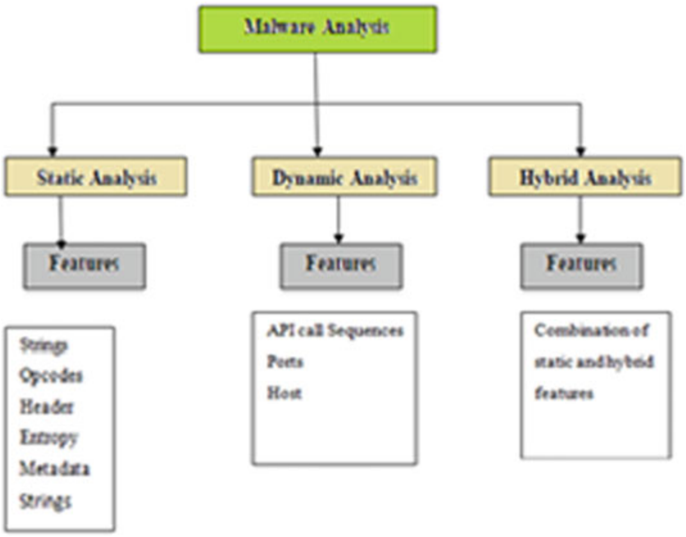


Fig. 3 Traditional malware analysis approaches

1.2.2 Features and Feature Engineering

The performance of any classification, prediction, and recognition system is closely dependent on feature. In machine learning, features are learned manually or we can say that hand-crafted features are used. They dominate the past on image- and video-based applications. There are many disadvantages associated with this feature learning like deep knowledge of data for feature extraction; feature extraction and classification were two different modules, where hundreds of features crafted for applications, feature dimension is high and to select optimized features from feature vector is a slow process.

Traditionally malware was identified and analyzed by utilizing the following approaches.

1.2.3 Static Analysis

It refers to the analysis or investigation of a malignant program without executing it. It is the process of extracting information from malware while it is not executing. Static analysis can be performed directly with the actual code (if present) and if not, can be applied to sundry representations of executables. Static analysis is considered the most facile, expeditious, and less precarious analysis process. It is the most facile and expeditious because there are no special conditions and requisites needed for the analysis process. The malware is simply subjected to analysis implements. It is less jeopardous because the malware is not executed during analysis; consequently, there is not at all any jeopardy of an infection yielding and spreading while analysis is going on, and we do not worry about engendering a safe environment for static analysis. The patterns detected in this kind of analysis include string signature, byte-sequence or operation codes (opcodes), frequency distribution, byte-sequence n-grams or opcodes n-grams, API calls, the structure of the disassembled program, etc. The terminus goal is to identify malware afore the program goes under assessment. Disassembly of malevolent programs is required to detect the patterns some prevalent disassembly implements are objdump, IDA Pro, etc. Static analysis is considered to be a less profit method of analysis as the data extracted from the static analysis is less promising because data is amassed when malware is in passive mode (not executing). Data extracted is constrained and not reveal much paramount information about malware. Prevalent techniques applied in the static analysis are flow analysis, string analysis and signature analysis.

1.2.4 Dynamic Analysis

Dynamic analysis is the process of extracting data from malware while it is executing. It refers to the analysis of the deportment of a malevolent program while it is being executed in a controlled environment (virtual machine, emulator, sandbox, etc) to identify inimical activities after the program executes. The demeanor is monitored by utilizing implements like process monitor, process explorer, wire shark, or capture bat. This kind of analysis endeavors to monitor system calls, injunctive authorization trace, function and API calls, the network, the flow of information, etc. Unlike the static analysis, which provides inhibited information from the malware being analyzed, the dynamic analysis offers an in-depth view into the malware's functions and comportment because it is accumulating information while the malware is executing. To conduct dynamic analysis we require two things, first is the environment where we can execute malware is in a controlled manner for the analysis purport and second is analysis implements that monitor and records the environment for any vicissitudes made by the malware to its target system. Unlike static analysis, dynamic analysis is considered to be highly jeopardous but paramount, or high-profit process. The peril of infection, spreading, or something inimical transpiring is high because the malware is executing; the profit is high because the data extracted from malware reveals more of itself during execution. In the dynamic analysis, we are probing for the following vicissitudes in registry activity, network traffic activity, process, and file activity. Some prevalent dynamic analysis implements are process monitor, wire shark, capture bat, anubis, etc.

1.2.5 Hybrid Analysis

The hybrid analysis technique includes consolidating static and dynamic features accumulated from examining the application and drawing data while the application is running, discretely. Nevertheless, it would boost the precision of the identification.

The principal drawback of hybrid analysis consumes the system resources and takes a long time to perform the analysis. The hybrid analysis amalgamates the traits of static and dynamic analysis for expeditious analysis and better results.

1.2.6 Comparison

Static analysis cannot detect unknown malware and its variants. Compared to static analysis, dynamic analysis is more efficacious and does not require the executable to be disassembled but on the other hand, it takes more time and consumes more resources than static analysis, being more arduous to scale. One more issue is as the controlled environment in which the malware is monitored is different from the genuine one, the program may comport differently because some deportment of malware might be triggered only under certain conditions such as via a concrete command or on the concrete system date and in consequence, cannot be detected in a virtual environment. In static analysis, data extraction is effective only if the malware is free from any type of encryption or obfuscation. Dynamic analysis is all about making the malware prosperously run in a controlled environment. Therefore, its circumscription is because of the different malware dependencies like time, event, program, etc. Static analysis can facilely be subjugated by a packed and encrypted file. This is why file unpacking and decryption are paramount in the fight against malware. Static analysis reveals some immediate information about malware but it is expeditious, exhaustive analysis more in-depth information but it is hard and time-consuming.

Malware analysis is a highly manual and laborious task, additionally requires analysts to have expertise in software internals and reverse engineering. Data mining and machine learning have shown promise in automating certain components of malware analysis, but these methods still rely heavily on extracting paramount features from the data, which is a nontrivial task that perpetuates to require practitioners with specialized skill sets. As the number of devices connected over the cyber world increases parallelly the attacks additionally increase exponentially. In reality, malware analysis does not reveal most of the information from the malware because of the known limitation of the malware analysis process.

1.3 Malware Classification

We now shift our discussion toward the main topic of this chapter that is malware classification and identification. In general, malware classification is defined as to group or classify malware together predicated on some mundane properties like they apportion homogeneous code, same potential damage, their inceptions, etc. In more simple words, classification is the process of assigning an object to a category or class. Classification refers to methods for presaging the likelihood that a given sample belongs to a predefined class or category, like whether a piece of email belongs to the class “spam” or a url is benign or malignant.

Malware can be classified in many ways such as depending on task, inception, authorship, damage potential, etc. In general, malware samples are grouped by family. Malware samples that show homogeneous functionality, structure with little differences are grouped under one roof and referred to as they belong to the same malware family. Classification is the prediction of incipient samples into its class whereas clustering is about discrimination of one group of samples from other groups. Classification is supervised whereas clustering is unsupervised. Classification examples are like to classify the taste of food as good or bad, to classify the thoughts or thinking as right and wrong, etc.

The prevalent term for non-maleficent files is a benign file. These are the examples of a binary classification problem one with only two output classes, “spam” and “not spam,” “botnet” or “benign.” By convention, samples that possess the attribute we are investigating (e.g., that an electronic mail is spam) are labeled as belonging to class “1” while samples that don’t possess this attribute (e.g., mail that is not spam) are labeled as belonging to the class “0.” These 1 and 0 class labels are often referred to as positive and negative cases, respectively.

Classification is a puissant and efficacious supervised learning model that can be applied productively to a broad range of security and other quandaries. The algorithms used to perform classification are referred to as “classifiers.” There are numerous classifiers available to solve binary classification problems, each with its strengths and impotencies. By the definition of malware classification, one can be confused with the identification of any given file as malicious and non-malicious. One should be kept in mind that malware classification includes the identification and classification of malicious and non-malicious files. So we can conclude that a given arbitrary binary file identified or classified as benign or malware comes under malware classification. This classification is utilized to determine whether a binary is malicious or not.

1.3.1 Classification Steps

A classification typically proceeds through the following steps:

1. A training/learning phase: In this phase, an analyst builds a model and applies a classifier on the training inputs. Training data consists of two things, data or samples, and its associated labels/class.
2. A validation phase: This phase is applied to assess the training performance on validation data. The validation phase is optional but researchers and analysts vigorously suggest utilizing the validation phase. In this, training data is split into two sets, one is for training and the second is for validation. Training is done on training data and to assess the training performance (customarily accuracy) we apply validation data on training.
3. A testing phase: To assess the performance of the deep learning model, we apply testing data on the classifier and monitor the classifiers prognosticated labeled

with an authentic or ground-truth label of test samples. The test precision is the overall precision of the model. test data is not optically discerned afore data.

1.3.2 Why Malware Classification?

Malware and its variant detection and classification have become one of the most adverse quandaries in the field of cybersecurity and the digital world. The daily increase of malware and its variants is a rigorous quandary of malware analysis. The main quandary with malware analysis is that the number of attack files submitted to antivirus companies for the investigation purpose is enormous. It is virtually infeasible and arduous to analyze each file manually, so there is a desideratum for some automation system and implements to analyze these files efficiently with less human intervention and efforts.

In the cybersecurity domain, traffic classification as malicious and benign is considered the first step toward security. By classifying malware into their respective families is helpful to analyze samples of a given family by human experts and some defensive measures can be proposed to mitigate malware attacks. Features or characteristics are extracted from the malware binaries utilizing data extraction methods and implements. The attack of malware and its variants is not only inhibited to the cyber world, it is withal affecting the IoT networks, mobile networks, and contrivances. Researchers and analysts commenced to explore malware analysis utilizing deep learning and visualization techniques in IoT, mobile, and cloud infrastructure.

1.3.3 Why Malware Visualization?

Malware visualization is the process of visualizing malware binaries as images—examples are given in Fig. 4. Visualization avails to visualize kindred attributes and distinctions between two variants of the same family. Visualization is efficacious in the representation of internal structure kindred attribute of malware. Malware binaries are ready to run or executable programs referred to as binary files and has an extension of .bin or .exe .

As we can visually perceive from Fig. 4 that malware from the same malware family exhibits the same internal structure while malware from different malware families has a different internal structure. This is the prevalent advantage of visual malware as an image and it avails in classifying malware. The advantage of images utilized in visualization is that they can give more in-depth information about the internal structure of the malware binary code and could identify even small changes in code while retaining the whole structure of the code.

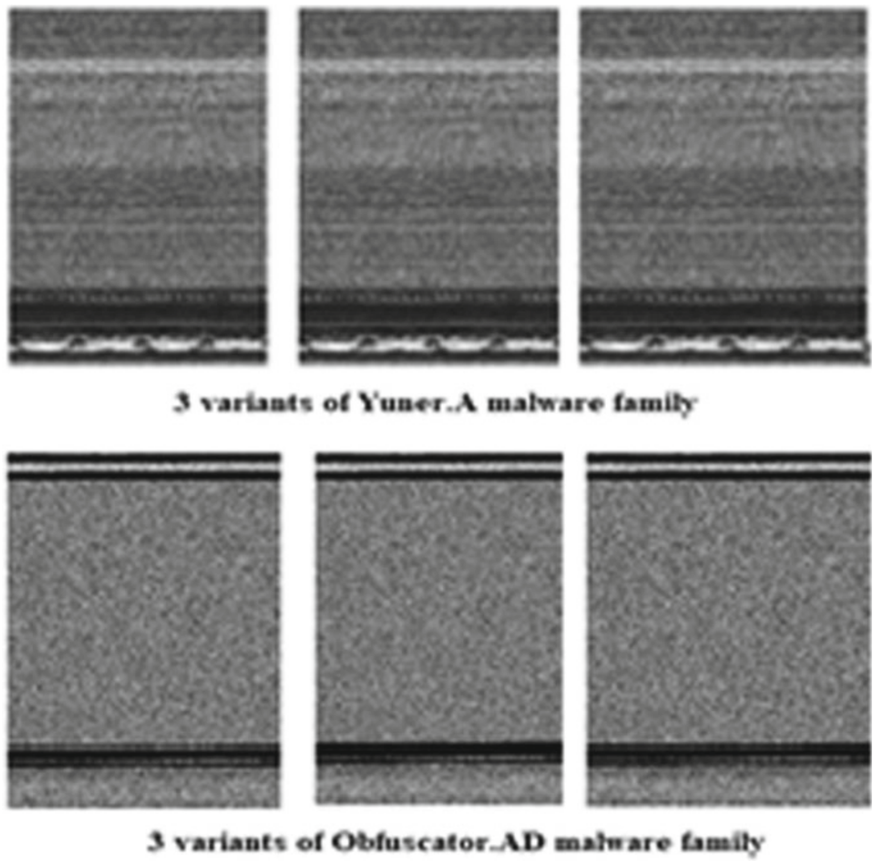

3 variants of Yuner.A malware family
3 variants of Obfuscator.AD malware family

Fig. 4 Visual representation of malware

1.3.4 Challenges of Malware Classification

Here we are going to discuss the challenges that are reported during the study of malware analysis. One of the most sizably voluminous challenges is that everyday millions of malware are being designed and the complexity to detect this massive amplitude of sophisticated malware are very difficult to identify. Traditional approaches for malware analysis were very tedious and manual intervention was required for analysis. Obfuscation techniques present most immensely colossal hurdle and one of the major factors which affect the analysis of malware. Scalability is one of the major challenges in the malware defense system as the number and variety of malware are kept incrementing. Classification algorithms and models can engender precise results on propitious conditions but this case is not possible in the genuine world. To obtain a dataset for training and testing that is sizably voluminous and accurately labeled is arduous. The number of samples in each class additionally affects the relegation precision. The classifier's performance is highly dependent upon the ample quantity of labeled data. Overfitting and underfitting are two well-kenned quandaries associated with the classifier's performance. There is not a single performance measure that is used to assess the performance of the classifier; there are varieties of measures available like accuracy, precision, f1 score, roc curves, etc. Deep learning is about deep neural networks and neural networks have a variety of hyper parameters that affects the models or classifier's performance like the number of hidden layers, number of neurons per layer, learning rate, dropout, etc. Some features extracted from malware samples have high dimensionality, which denotes a more involutes system and incremented processing time. One of the latest emerging threat in malware analysis is the file less malware [10], it does not utilize the file system for its malevolent activities, thereby eschewing traditional approaches and became one of the hurdles in malware analysis.

2 Deep Learning

In this section, we are discussing what is deep learning, what are different deep learning algorithms, how is the deep learning model defined? The topic is briefly explicated to relate with the malware classification.

2.1 What is Deep Learning?

Deep learning is a sub-branch of machine learning and its functioning is inspired by the structure and function of the brain called neural networks. Deep learning refers to the set of techniques utilized for learning in neural networks. It refers to deep, or many-layered, neural networks withal kenned as deep neural network. Deep learning is about learning abstract representations of data or observations utilizing network layers that avail to make sense of some kind of hidden patterns, features of data like images, sound, and text. In pursuing malware analysis and lowering human intervention, deep learning has been introduced into malware analysis. Deep learning depends on studying various levels (from low level to higher level) of representations, where top-level features (for example, face) are tenacious from lower level ones (like edges, curve, etc.), and similarly lower- level features avail in determining numerous top-level features.

2.1.1 Machine Learning

Machine learning is defined as the subfield of artificial intelligence. The goal of machine learning is to understand the data and build a numerical model, fit that data into a model that can be understood and utilized by the user.

Antivirus companies commenced to utilize modern classification techniques dependent on data mining and machine learning methods. All the methods either data mining or machine learning approach dependent upon the extraction of features, applying more clever frameworks or classifiers for classification purposes. The disadvantage of machine learning is that it requires manual feature extraction. Many authors applied support vector machine (SVM) classifier, naÃ¯ve bayes classifier, or mixed classifiers to classify malware.

2.1.2 Shallow and Deep Learning

Deep learning is a subfield of machine learning, concerned with functionality and structure inspired by the human brain called artificial neural networks. The term "deep" in deep learning isn't a reference to any kind of in-depth understanding achieved by the approach; rather, it stands for conception and number of stacked layers of representations of the input. How many layers contribute to a deep learning model of the input data is called the depth or deepness of the model? The term shallow learning algorithms are normally referred to as traditional machine learning algorithms. It refers to algorithms that are not deep in architecture, e.g., decision trees, support vector machines, naive bayes classifier, etc.

Modern deep learning models often constitute tens or even hundreds of stacked layers of representations and they're all learned /extract features automatically from exposure to training data. Machine learning inclines to fixate on learning/extracting only one (mostly) or two layers of representations of the input data; hence, they're sometimes called shallow learning.

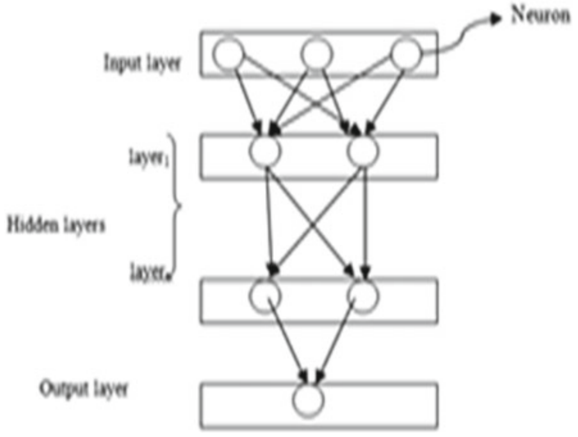
2.1.3 What Makes Deep Learning Different?

1. Deep learning algorithms offered better performance on many involutes real-world problems.
2. It makes problem-solving more facile.
3. It automates the most critical phase of machine learning that is optimized feature extraction.
4. With deep learning, we can acquire more refined transformations of complex problems.

2.1.4 Deep Learning Framework

Generally, a framework is a platform, interface, accumulation of libraries, and implements for developing applications. We have deep learning frameworks for building deep learning models facilely and without going into depth cognizance of algorithms. Some popular frameworks are tensor flow, keras, pytorch, caffe, deeplearning4j, etc.

Fig. 5 Deep neural network



2.2 *Deep Learning Algorithms*

Deep learning can be considered as a subfield of machine learning. It is predicated on learning and improving on its own by examining algorithms. While machine learning uses simpler concepts, deep learning works with artificial neural networks, which are designed to be homogeneous to how humans think and learn. Artificial neural network (ANN) drive deep learning. Neural networks were restricted by computing power and thus were limited in complexity. However, deep learning (ANN with many layers) sanction computers to observe, learn, and react to intricate situations more expeditious than humans. Deep learning has availed image classification, language translation, and speech recognition. Deep learning can be acclimated to solve any pattern recognition problem, to classify images, for language translation, to recognize speech and without human intervention. Deep learning is to learn hierarchical representations of input data.

Commonly used deep learning algorithms are

2.2.1 Deep Neural Network (DNN)

Deep neural networks are the ANN with many layers Fig. 5. Typically deep neural networks are feed-forward networks in which input flows from the input layer to the output layer and hidden layers(two or more) and the sodalities between the layers are one way which is in the forward direction(input layer to output layer). The outputs are obtained by learning with datasets of labeled information predicated on backpropagation. The circumscription of deep neural networks is that they don't have any memory unit.

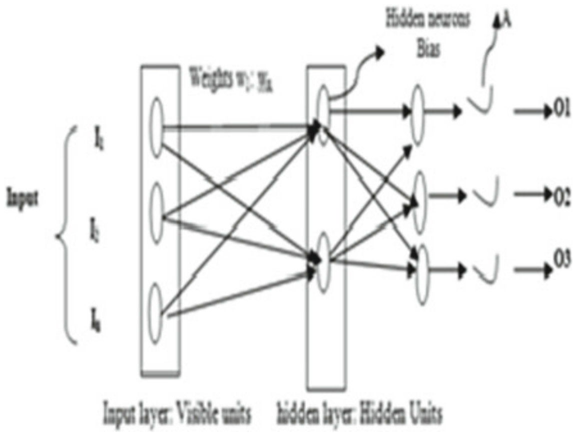


Fig. 6 Restricted boltzmann machine

2.2.2 Restricted Boltzmann Machine (RBM)

RBMs are a two-layered artificial neural network with generative capabilities Fig. 6. They can learn a probability distribution over its set of input. RBM can be utilized for dimensionality reduction, relegation, regression, collaborative filtering, feature learning, and topic modeling. RBMs are a special class of Boltzmann machines and they are restricted in terms of the connections between the visible and the hidden units. This makes it facile to implement them when compared to boltzmann machines. As stated earlier, they are a two-layered neural network (one being the visible layer and the other one being the hidden layer) and these two layers are connected by a fully bipartite graph. This denotes that every node in the visible layer is connected to every node in the hidden layer but no two nodes in the same group are connected. There are two other layers of bias units (hidden bias and visible bias) in a RBM. This is what makes RBMs different from auto encoders. The hidden bias RBM produces the activation on the forward pass and the visible bias avails RBM to reconstruct the input during a rearward pass. The reconstructed input is always different from the actual input as there are no connections among the visible units and therefore, there is no way of transferring information among them.

2.2.3 Convolutional Neural Network (CNN)

Convolutional neural networks are very subsidiary for images based processing, especially for image-based classification. A convolutional neural network Fig. 7 is a type of feed-forward neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex, whose individual neurons are arranged in such a way that they respond to overlapping regions tilling the visual field. Convolutional layers are the core of a convolutional neural network. Convolutional neural networks, like neural networks, are composed of neurons with

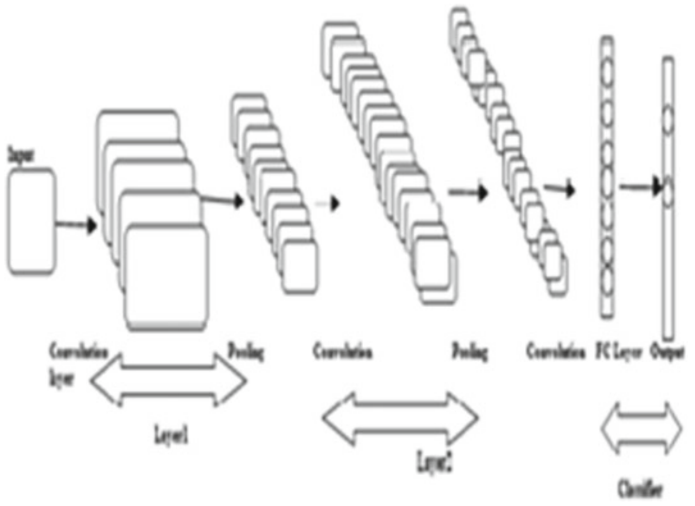


Fig. 7 Convolutional neural network

weights and biases (updates through learning). Each neuron receives inputs, applies a convolution operation (weighted sum of multiplication) over them, passes it through an activation function, and responds with an output. The network has a loss function and weights and biases are updated according to the loss function. CNN is composed of three types of layers: convolution layers, pooling/subsampling layers, fully-connected/dense layers.

2.2.4 Deep Belief Network (DBN)

A DBN is a class of deep neural network, a graphical model, composed of multiple layers of latent variables (hidden units utilized for detecting features), with connections between the layers but not between units within each layer and have direct and undirected connections Fig. 8. RBMs can be stacked and trained to compose so-called deep belief networks. Multiple RBMs can withal be stacked and learned through the process of gradient descent and backpropagation. Such a network is called a deep belief network. A deep belief network utilizes an unsupervised machine learning model to produce results. One of the mundane features of a deep belief network is that albeit layers have connections between them, the network does not include connections between units in a single layer. A DBN can work as a supervised learning algorithm (as a classifier) and additionally utilized as an unsupervised learning algorithm (to cluster data).

Fig. 8 Deep belief network

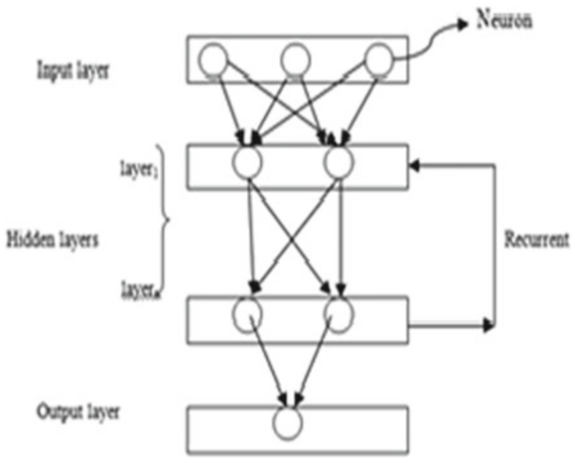
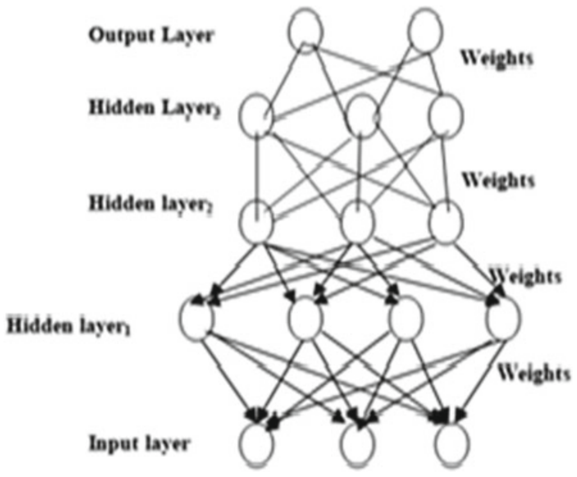


Fig. 9 Recurrent neural network



2.2.5 Recurrent Neural Network (RNN)

Recurrent neural networks are best to process sequences. A recurrent neural network Fig. 9 addresses the issue of the memory limitation of deep neural networks. Deep neural networks are stateless, but recurrent neural networks have connections between passes and connections through time. A recurrent neural network looks similar to a traditional artificial neural network except that it has a memory-state and is added to the neurons. With a recurrent neural network, this output is sent back to the previous layer number of times. RNNs can remember parts of the inputs and use them to make accurate predictions.

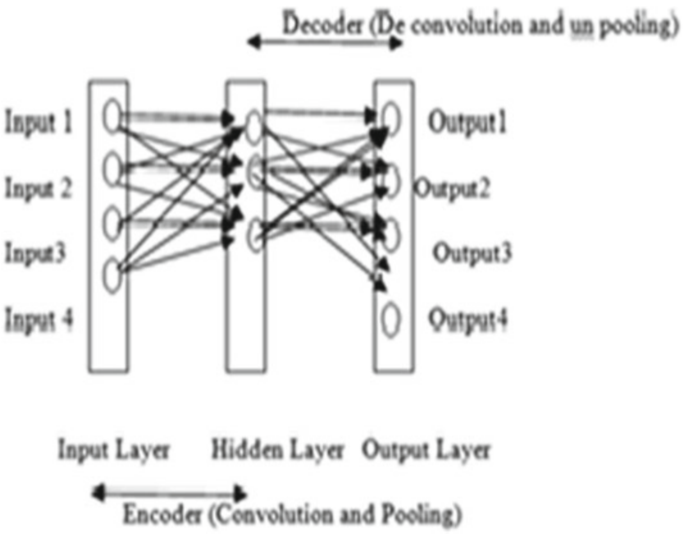


Fig. 10 Autoencoder

2.2.6 Deep Autoencoder (AE)

Autoencoder is a neural network Fig. 10 that utilizes unsupervised learning algorithms and backpropagation. It efficiently compresses and encodes input data then learns how to set output values identically to the input values. How to decode the data back from the minimized encoded representation to a representation that is as proximate to the original input as possible. Autoencoder, by design, transforms data into a hidden representation and then reconstructs data from that hidden representation inputs are high-dimensional data. It is compressed by the hidden layer and the output layer reconstructs the inputs. The main applications of the autoencoder are data denoising and dimensionality abbreviation.

2.3 Steps for Building a Deep Learning Model

The main advantage of deep learning systems for malware analysis is that they automate the work of feature extraction, and they have the potential to perform more accurately and efficiently than traditional approaches to malware analysis, especially we want to focus on malware classification especially on new, previously unseen malware. Essentially, the following steps Fig. 11 are used to build any deep learning model for malware classification.

1. Data/samples collection: To train the DL model, we require data (training data). For malware analysis, we require malware as well as benign (good wares) data.

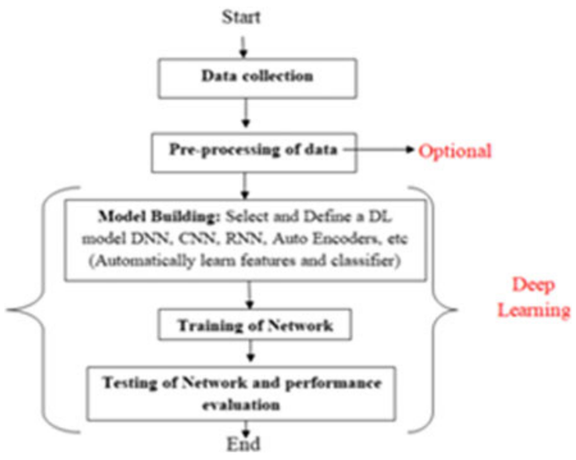

Start
Data collection
Pre-processing of data
Optional
Model Building: Select and Define a DL model DNN, CNN, RNN, Auto Encoders, etc (Automatically learn features and classifier)
Training of Network
Testing of Network and performance evaluation
Deep Learning
End

Fig. 11 Deep learning model building

The performance of the dl model depends profoundly on the quantity and quality of training examples, you provide for training. The quality of the training data is also important. If you want to apply the dl model for a multi-class classification problem you have to amass adequate data for each class. The general rule of thumb is that the more data (training data) you feed into your dl model for training, the more precise results you will get.

2. Model building: We have to define a deep learning model among various available deep learning models (DNN, CNN, RNN, Auto Encoders, etc.) as per the requirements. We first build the model then training and testing is applied on the defined model.
3. Training: Train the model for recognition of malware on the optimized features extracted automatically by the dl model. For training, we provide data /samples and associated labels of samples. Mundanely, training is considered to be an arduous task to perform because of the settings of hyper parameters. We feed the training images into different CNN model architectures (it varies with several layers, number of neurons in layers, learning rate, number of epochs, batch size, etc.) with different hyper parameters settings, several epochs, and batch size and probe for the model that fits our dataset.
4. Testing: Once you trained your model, we require to test the model on the data samples that were not included in the training to assess the model's performance or how precise the model is. Generally, testing is done by running the trained deep learning model on the data samples that were not included in the training denotes data that has been never seen by the model.

3 Malware Classification Based on Malware Visualization and Deep Learning

In the following section, we are going to present and discuss detailed procedures on the recent cognate work predicated on malware classification utilizing deep learning. The exhaustive study covers techniques of malware visualization predicated on different deep learning algorithms.

To visualize malware as an image [19] is the field of representing malware in the form of visual features. To analyze malware more deeply, malware has to be transformed into an image, refer Fig. 13. The main benefit of visualizing malware as an image is that different sections of a binary file can be facilely differentiated, Fig. 4. Many solutions have been proposed and implemented utilizing static and dynamic approaches but this work is predicated on the malevolent code and its variants detection and classification utilizing visualization techniques and deep learning. There has been extensive research and study done on analyzing malware, many papers are published which denotes static, dynamic, and signature-based malware analyzing techniques. A publication on image-based malware visualization is one of the preferred ways [19]. This section explicates how to compose an image out of binary malware files, how to visualize those images, and how these images are utilized for image-based classification.

Traditionally this task is done by signature matching. In signature matching, a database is prepared of properties, the behavior of previously seen malware; incipient binaries are compared by this dataset to compare previously stored data to determine that something visually perceived afore. Signature matching performs well as long as malware designers alter the behavior and properties of malware to evade detection. A malware designer continuously changes the properties and behavior of malware to avoid detection. By utilizing obfuscation techniques like metamorphic and polymorphic, authors of malware changes properties of code, behavior to avoid detection of malware by signature matching or malware identification implementations.

We have studied papers which utilize the same principles as [19] to classify the malware into their families. It has been observed that the deep learning model is efficient. We propose to utilize malware visualization technique, converts every malware bytes code to a grayscale image. In research and analysis, it was observed that malware from different families has kindred attributes in visual appearance presenting to us an opportunity to exploit this impotency where these images will be utilized for image-based classification. In image generation and classification technique, every byte of data is converted into a grayscale pixel; array of the byte stream was converted into an image. Image representation of the malware engenders very convincing images for analysis purposes.

3.1 *Related Work: Recent Innovations in Malware Classification Using Deep Learning and Visualization*

In this section, we are going to discuss and review the current state-of-the-art approaches that have been established to address the malware classification utilizing deep learning models and malware visualization.

The solution based on malware visualization by Nataraj et al. [19] in the year 2011 is considered as the first solution of this kind. Authors proposed a method to represent malware as a grayscale image and after that extracted gist (a texture-based feature), afterward, the grayscale malware images are classified utilizing a data mining algorithm knn (k-nearest neighbor). They experimented with the malimg dataset consisting of 9458 malware grayscale images belonging to 25 different families, amassed from the anubis system. For the experimental purpose, they converted malware into a grayscale image of dimension 64*64. They extracted 320-dimensional texture features from malware image predicated on gist. They divided the samples into a ratio of 90–10% for train and test ratio. They obtained the test accuracy of 97.18% which is very high as compared to traditional approaches. The results obtained evidence that visualization of malware is very efficacious and can relegate malware with more precision and expeditiously than subsisting static and dynamic approaches.

In the year 2013, Han k. et al. [5] proposed an incipient way of visualizing malware. They visualized malware as a color image utilizing binary values. The proposed method generates RGB colored pixels by utilizing binary information extracted through static analysis. First, the author disassembled malware binary files utilizing implements such as IDA pro or ollyDbg, after disassembling the extracted sequence of assembly codes are divided into blocks of opcodes (example of opcode sequence: pusmovaddsubmov), after block building, every block of opcode instruction sequence is processed by two hash functions to engender matrix of coordinate values and RGB color pixels information. To compute the homogeneous attribute between image matrices authors utilized a selective area matching algorithm. For experiment purposes, the authors utilized a color image of size 256 * 256, 2505 benign, and 8169 malware image matrices are engendered utilizing a visualization implement. 95% test accuracy is achieved by this method. Results deduced relegation efficaciously and the time spent to calculate homogeneous attribute was about 2.4ms.

In 2016, K. K. Pal and k. S. Sudeep [21] presented a data preprocessing technique for the malware relegation model utilizing a convolutional neural network and image representation. Authors proved that by applying preprocessing techniques on the data, classification accuracy can amend. Raw data applied to any deep neural network does not engender good results. The authors conducted three types of normalization on the dataset and showed how precision varies. They applied to mean normalization, standardization, zero component analysis on the dataset. For experimental purport authors used malware color images of size 32*32 and they utilized the cifar 10 dataset (dataset contains 60000 color images of size 32*32 belongs to 10 different classes). They obtained an accuracy of 64–68% when zero component analysis is applied, they got increased accuracy as compared to when no preprocessing applied.

In the year 2016 Ding y. et al. [2] have prosperously applied a deep belief network, one of the unsupervised learning algorithms for malware relegation. The authors represented malware as opcode sequences and then use deep belief network to detect malware. An opcode (operation code) additionally kenned as an instruction machine code that designates the operation to be performed. In a deep learning algorithm, the neural network is trained multiple times by the raw opcode sequences extracted from the decompiled file, so that the hidden feature information can be efficaciously learned and the malware can be detected efficiently and more accurately. Feature extractor measures different measures like information gain, document frequency to evaluate the relegation . Author used information gain to cull subsidiary n-gram. To accurately describe the opcode comportment, the author extracted opcode sequences from 3000 benign and 3000 malware samples. The extractor evaluates 10000 different n-grams with different information gain values. From these 10000 values author utilized the top 400 n-grams as the features of an executable. DBN architecture has 3 hidden units with 200,200 50 hidden neurons, respectively. Each layer is trained with 30 epochs. The authors obtained 96.1% accuracy.

In the year 2016 Hardy w. et al. [6] proposed an intelligent deep learning framework for malware detection. They applied auto encoder, it is one of the unsupervised deep learning algorithms used to detect generic features from malware to detect unknown malware. They utilized a greedy-based feature learning at each layer, followed by supervised tuning of weights and biases. The authors extracted windows-based API call sequences from the portable executable (PE) files. For experimental purpose, authors used the comodo cloud security center dataset (dataset contains 22500 malware samples and 22500 benign samples total of 50000) and the train and test ratio was 90–10%. The experiment is performed with a different number of neurons in the hidden layer but 100 neurons at each hidden layer and 3 hidden layers configuration yield the maximum accuracy that is 96.85% at training and 95.64% at testing.

Tobiyama s. Et al. [23], in the year 2016, proposed the fusion of deep learning models in malware analysis. The authors first applied a recurrent neural network to extract the features based on malware behavior and then applied CNN to classify malware feature images of size image 30*30. To capture the behavior of malicious application authors utilized API call sequences. The proposed malware detection framework is mainly using API call sequence extraction and deep learning technique for classification. A process behavior is defined as various activities and to perform each activity various operations are associated with activities. To record process behavior API call sequence is generated; the API call sequence represents activities and related operations. They extracted feature vector by training of recurrent neural network and then these extracted feature vectors are converted into an image and applied CNN for classification. For experimental purpose 81 malware process log files of 11 different malware families, 69 benign processes log files data collected by NTT secure platform laboratory. The architecture of recurrent neural network consists of an input layer, a hidden layer, 2 LSTM hidden layers, and an output layer. The architecture of CNN consists of 2 convolution and pooling layers with 10 and

20 filters, respectively. Used max pooling with stride 2, no of epochs: 5, batch size: 20. They obtained 96% accuracy.

Azab A. et al. [1] in the year 2016 proposed and addressed machine learning technique for identification of untrained botnets traffic. Authors applied the c4.5 learning algorithm (for building classifier with 10,20 and 30 FN costs and 1 FP cost) and correlation-predicated feature cull (cfs, applied to filter out duplicate, redundant and impertinent features form extracted features) algorithm on the communication traffic between compromised contrivances and botmaster, they extracted 511 different features coalescence from 9 different features categories from this communication that avails to relegate between botnet traffic and legitimate traffic. For botnet network traffic accumulation, Zeus (a botnet toolkit) was culled and it is considered one of the major threats, especially for attacks on online banking transmissions. Two separate datasets accumulated for experiment one are for training (the 432 botnet traffic engendered utilizing zeus builder version 1.x and 2774 HTTP traffic) and the second is for testing(the 144 botnet traffic engendered utilizing Zeus builder version 2.x and 2396 HTTP traffic). All the built classifiers were evaluated utilizing the K-10 cross-validation to optate the lenient classifiers. The built classifiers were evaluated utilizing the K-10 cross-validation to cull the rigorous classifier. The voting results from the three costs achieved 88 TP, 56 FN, and 1 FP results, providing 0.989 precision, 0.611 recall, and 0.755 F-Measure results. These results betoken that the utilization of the stringent classifier might affect the detection of the untrained version's flows that were included by the lenient classifier.

In the year 2018, Kalash M. et al. [11] proposed and implemented a deep CNN model for malware classification. They translate the malware classification problem into an image classification by following the approach used by Nataraj et al. [19], converting malware binaries to grayscale images of size 224*224 and then applied a convolutional neural network for classification. The proposed convolutional neural network model architecture is based on VGG-16. They applied the proposed method on two different datasets, namely, malimg(dataset consists of 9458 malware samples belonging to 25 different families) and Microsoft dataset (contains 21741 malware samples, each malware sample belongs to 9 different malware families). Train and test ratio used by the authors are 90–10% in the malimg dataset and 10868 samples for training and 10873 samples for testing on the Microsoft dataset. They utilized cross-entropy loss to train the network. The authors achieved 98.52% accuracy on the malimg dataset (with 25 epochs and a batch size of 6) and 98.99 and 99.97% on two different settings of Microsoft dataset (with 25 epochs and a batch size of 8).

In 2018, Ni S. et al. [20] proposed a malware classification algorithm that utilizes static features and convolutional neural network. They converted the disassembled malware codes into grayscale images based on simhash, and then classification is done by convolutional neural network. They extracted the opcode sequence from the code section as features then after extraction of the opcode sequence they calculated simhash for sequence similarity comparison. By using simhash and bipolar interpolation they converted the opcode sequence into a malware image then applied convolutional neural network for training and classification. Each input image needs to go through two convolutional layers, two subsampling layers, and three full connection layers. During the convolution process, they applied 32 filters of size 2*2 and during subsampling max pooling is used whose size is 2*2 to dimension reduction. The authors used the dataset for the experiment in Microsoft malware classification challenge on kaggle by Microsoft 2015. The dataset consists of 10868 labeled malware images from 9 families, from 10868, 80% of them used for training and the rest for testing. The classification accuracy they obtained was 99.260% with a 98.07% f1 Score and 2.34% false positive rate (FPR).

Kim C. H. et al. [13] in the year 2018 proposed a convolution gated neural network for the task of malware identification and classification. Proposed model comprised of convolutional neural network, gated recurrent unit (GRU), layer of deep neural network, and a sigmoid layer. Each convolutional neural network has a convolution layer, activation function, and pooling layer. All convolutional neural network produces a single output, and this output is applied to gated recurrent unit layers and treats this output of convolutional neural network as time-series data. Each gated recurrent unit produces a single output equal to the number of convolutional neural networks in the first layer. Output of GRU is input to deep neural network. Each deep neural network produces single output. The final layer of the network is the sigmoid layer and the result of this layer is the classification.

In the year 2019, Singh A. et al. [22] explored and implemented a new way to represent malware as color images as they used RGB representation of malware (RGB images of size 32*32) over grayscale images to classify malware. They experimented with 37374 binary samples belonging to 22 families collected from malshare, virusshare, and virustotal, and malimg dataset. They applied deep neural network architectures ResNet-50(residual network) architecture including a dense convolutional neural network for classifying images. With their implemented model they obtained 98.98% using convolutional neural network and 99.40% using ResNet-50 on the authors dataset and 96.08% using convolutional neural network and 98.10% using ResNet-50 on the malimg dataset. The authors introduced a novel approach to convert the binary file string of zeros and ones into rgb color images. They used a 15 layer convolutional neural network model (5 convolutional layers and 2 dense layers).

Yin Q. et al. [25] in the year 2019 presented a fused model of convolutional neural network and recurrent neural network for image classification. Authors extracted features using convolutional and recurrent neural network networks from the intermediate convolutional neural network network.

In the year 2019, Naeem H. [17] proposed a fast deep learning model to detect malware in the IoT network. IoT devices improved the user experience of the internet by smart devices to connect and information sharing. The author proposed the detection of malware by converting malware binaries into the color images of size 192*192 and then applied a deep convolutional model for efficient malware detection on the malimg dataset (dataset consists of 9458 malware samples belonging to 25 different families) and leopard mobile datasets(contains 14733 malware samples and 2486 benign samples of IoT applications.) The train and test ratio was utilized as 55–45% for the malimg dataset and 34–66% for the leopard dataset. The author obtained an accuracy of 98.18% on the malimg dataset and 97.34% on the leopard dataset. The author achieved better accuracy and response time.

In the year 2019, Khan U. R., et al. [12] defined an improved, more intelligent convolutional neural network model for intrusion detection. Authors mentioned that machine learning algorithms have a low detection rate, as well as manual extraction of features, which is a laborious and time-consuming task that's why they applied deep learning in intrusion detection. Deep convolutional neural network is used for training and classification and it automatically extracts optimized features from input samples. The dataset used for experiments is the KDD99 dataset; the dataset contains 494021 training samples and 311029 test samples, from 5 different categories (contains normal, DOS, R2L, U2R, probe). They obtained the accuracy on improved convolutional neural network that is 99.23 for 800 epochs, which is a promising result. CNN model architecture has two convolutional and two pooling layers.

Mourtaji Y. et al. [16] in the year 2019 proposed a deep learning framework for malware classification. Authors first converted malware binaries into grayscale images as used by Nataraj et al. [19] and then trained a convolutional neural network model for classification. Different parameters for experimental purpose used malimg and Microsoft datasets. Train-test ratio used 85–15% for the malimg dataset and 10868 samples for training and 10873 samples for testing, and they utilized the convolutional neural network architecture defined by K. Simonyan and used a cross-entropy to learn and train the model from the network after that utilize stochastic gradient descent (SGD) to optimize the learning parameters of the model, initialized the learning rate to be 0.001 and 25 epochs, batch size of 6 for malimg dataset and 25 epochs, batch size of 8 for Microsoft dataset. The authors obtained 97.02% on malimg and 98.72% and 99.881% on two different experiment settings on Microsoft dataset.

Jain M. et al. [9] in the year 2020 applied and compared CNN and extreme learning machines (ELM) for malware classification. Results are evident that ELMs required less time to train as compared to train a CNN and achieves higher accuracy on one-dimensional data processing. Authors also found that for two-dimensional data processing ELMs are faster than CNN. Authors experimented with different settings of the CNN model like they applied CNN with one hidden layer than with two hidden layers with different hyperparameters settings, and the best results they got with a two-layer configuration of CNN with input images of size 128 $\times$ 128 pixels with 32 and 64 filter maps. With ELMs, they have to perform very fewer experiment settings like only they tuned some neurons in the hidden layer, the chosen 50 neurons for the experiment. The authors utilized grayscale images of size 128*128 for CNN and grayscale images of size 64*64 for the ELM model. They used the malimg dataset for experiments and 80% for training, 10% for testing, and 10% for validation division is applied to the dataset. The configuration of CNN architecture: CNN with two convolutional layers, 128 $\times$ 128 images, and (32, 64) filters and ELM architecture has 50 neurons in the hidden layer. They obtained an accuracy of 96.3% on the CNN model and 97.7% on the ELM model.

In January 2020, Kumar G. S. and Bagane P. [3] presented a hybrid deep learning-based model for malware classification. They applied convolutional neural network with bi-directional long short-term memory(LSTM) to do the task. First, they applied convolutional neural network for feature extraction, and then in the last layer after flattening the output they applied the LSTM model for the classification.

In the year 2020, Vasan D. et al. [24] proposed a novel approach based on the ensemble CNN architecture model for effective detection and classification of malware images. Authors utilized the pre-trained models and combined different optimized features extracted to fine-tune the VGG 16 and ResNet50 and fused the extracted features from both models and classified the malware into their corresponding families. Results proved the effectiveness of the proposed method.

In the year 2020, Naeem et al. [18] proposed a deep learning model for malware detection in the android operating system. They transformed a raw android file into a color image (of dimension 224*224 and 229*229) and then applied a deep convolutional neural network model for android malware classification. The author designed a very deep convolutional neural network that has 4 convolutional layers each followed by an activation function and max pooling layer, followed by a dense layer and softmax layer. The author applied a deep convolutional neural network model on the leopard dataset (dataset contains 14,733 malware samples and 2486 benign samples of different IIoT applications) and the malimg dataset. They achieved 97.81% accuracy on a leopard mobile malware dataset (224*224 color image dimension), a well-known industrial Internet of Things (IIOT) dataset, and 98.79% on a malimg dataset (with 229*229 color image dimension).

3.1.1 Generative Adversarial Networks (GANs)

Generative adversarial networks provide a new way of addressing computer vision, detection and classification problems. One of the biggest problems with deep learning model is lacking of sufficient training samples as we know that good quality and sufficient data is the key of deep learning model. Any deep learning model heavily dependent on the number of samples providing for training. Many datasets available today face this problem. We can notice from Table 1 that malimg dataset is also a high imbalanced dataset Allaple. A malware family contains 2949 samples as compare to Wintrim.BX and Skintrim.N malware family contains 97 and 80 samples respectively. This imbalance affects the training process as well as the classification performance. To address this problem GANs we can utilize. GAN can be used to generate samples from the data.

GANs are types of deep learning technique for generative modeling and most recent development in machine learning. GANs are very incipient in the literature on deep learning, and they belong to unsupervised learning. The first paper published by Goodfellow et al. In 2014 [4] introduced the generative adversarial networks framework. A GAN is trained utilizing two neural network models. Generative modeling requires a model to engender incipient samples from a subsisting distribution of available samples, for example, engendering incipient images that are generally homogeneous but concretely different from available images in the dataset. GANs are mainly utilized with convolutional neural network which denotes GANs are specially utilized for image cognate applications. A GAN is trained utilizing two neural network models. One model is referred as the generator or generative network model, which learns to engender incipient likely samples. The other model is called the discriminator or discriminative network and learns to differentiate engendered samples from authentic samples, discriminator works like a classifier; it distinguishes authentic samples from the engendered samples.

We can apply GAN in cybersecurity field, and it is proving very promising. One of the quandaries with any deep learning model is data imbalance issue. As we all very well know that good quality and adequate magnitude of training data is the key for any deep learning models performance. So, we can surmount this issue of data imbalance utilizing GAN and can engender incipient samples from the genuine samples for training. In reality, all the real-world datasets are imbalance datasets and there is much variation in the number of samples in each family. So GANs can be proved very efficient to address this issue.

In the year 2019, Y. Lu and J. Li [15] applied GAN for malware classification/predication on deep learning model. Authors addressed the data imbalance issue and engender incipient samples for training. They applied GAN utilizing convolutional neural network and called this model as deep convolutional generative adversarial network (DCGAN) to engender malware samples from the available dataset. Experimental results are conspicuous that utilizing GAN accuracy of the proposed model is incremented by 6%. In their implementation, they utilized a 18-layers deep residual net as the malware classifier. Network learns from the trained data that engenders the potential distribution of the incipient genuine samples from the authentic samples, while the discriminator differentiates the incipient genuine samples with the genuine samples as accurately as possible. Multiple convolutional and convolutional-transpose layers are utilized in the discriminator and engenderer for training. They trained the GAN network for 10000 epochs to engender the authentic samples, starting from the 1000 training epochs, preserved 25 engendered samples for every 100 epochs for each class. So after the training is done. They have 2250 engendered synthetic samples for each class. They achieved the overall average testing accuracy of the deep residual network is 84% and the precision, recalls, and f1-scores of the classes with more samples size are supplementally incremented.

In the year 2017, Kim JY. et al. [14] proposed a transferred generative adversarial network (tGAN) for automatic malware relegation and detection of the zero-day attack. They surmount the constraint of GAN training to pre-train GAN with auto encoder structure. The proposed model gets the best performance compared to the conventional learning algorithms. To address the data imbalance issue and to engender incipient samples they proposed and applied tGAN model predicated on GAN. Their proposed architecture consists of three modules: pre-training module, engendering data module, and malware detecting module. First module pre-trains the second module which has an engenderer that engenders data kindred for training, and a discriminator that distinguishes genuine data from engendered data. The discriminator is trained to distinguish the authentic data from the engendered data, and the engenderer is trained to make the discriminator to classify the engendered data into the genuine data. They used malware data utilized in the kaggle Microsoft malware classification challenge. The accuracy of malware type detection is 96.39%. The entire data is divided into training and test data at a ratio of 90:10. It shows the best performance compared to other conventional models, and it enables to detect malware even with a minute of data.

A detailed review of generative adversarial networks and its application in cyber security is presented by Banjo Y. et al. [26]. They explained how GANs are very useful and applicable in cyber security field. They reviewed two very widely utilized GAN architectures the deep convolutional generative adversarial network (DCGAN), and wasserstein GAN. Their reviews are notable to study cyber security where the GAN plays a vital role in the design of a security system . This paper guide the scope of modern cyber security studies with generative adversarial networks.

Deep learning can help to solve problems caused by modern malware and the way they function. Using deep learning we can also automate the malware analysis process. The biggest advantage with deep learning is that the manual extraction of features or data is skipped, deep learning architectures automatically extracts features from samples, based on the extracted features from the training dataset, samples are distinguished by samples belonging to a particular class to other classes. Traditionally malware classification has been a manual process, involving experts having in-depth knowledge of malware, their working, properties in malware to design malware identification, or classification engines. Deep learning facilitates automatic extraction of optimized features from the training dataset, letting the automatic detection of features analysts can make effort for designing more efficient algorithms, and better results.

3.2 Performance Metrics: To Measure the Performance of the Deep Learning Model

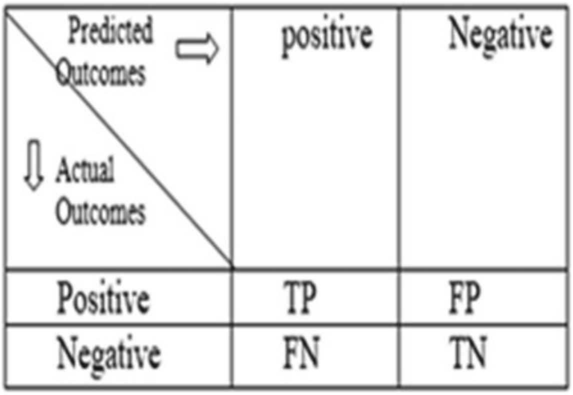
To evaluate the performance of the developed system or solution following metrics are calculated. Using these metrics we can compare different techniques and can conclude which technique is better than others.

3.2.1 Confusion Matrix

It is utilized to visualize the performance of a technique. In general, a classifier is evaluated by a confusion matrix Fig. 12. Structure-wise confusion matrix is a table representation that is used to describe the performance of a classification model on the test datasets. All other performance metrics are calculated utilizing the confusion metric. In the confusion matrix, there are four possible states denominated true positive (TP), false positive (FP), true negative (TN), and false negative (FN) defined as follows

TP: when the sample is identified as an attack and the sample is an attack (Remark: identification of attack).

Fig. 12 Confusion matrix



Predicted Outcomes ⇨ / ⇩ Actual Outcomes	positive	Negative
Positive	TP	FP
Negative	FN	TN

FP: when the sample is identified as an attack and the sample is not an attack (false alarm).
TN: when the sample is not identified as an attack and the sample is not an attack.
FN: when the sample is not identified as an attack and the sample is an attack.

Accuracy indicates the proportion of all samples with correct predictions to the total sample size. The formula to calculate accuracy is

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

Precision describes the ratio of predictive positive samples positive. The formula to calculate precision is

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

Recall is also known as True Positive Rate (TPR.)The formula to calculate recall is

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

F1 is the harmonic mean of precision and recall. The formula to calculate the F1 score is

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Receiver operating characteristic curve (ROC) is a graph that is used to summarize the performance of a classifier over all possible thresholds. The graph is generated by plotting a graph between True Positive Rate (TPR) and False Positive Rate (FPR). the formulas for TPR and FPR are

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \text{ and } \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}.$$

We observed that accuracy (1) is a common measure used to judge the classifier's performance but it seems inadequate, and other measures like f1-score (4) and recall (3) are also important to evaluate the performance of the classification. High accuracy and recall with lower misclassification are required for an efficient model.

3.3 A Practical Implementation of Malware Classification Using CNN and Malware Image Visualization

In this section, we are going to discuss a practical example of malware classification utilizing a convolutional neural network, which is considered the most prosperous deep learning architecture in computer vision, pattern matching, and natural language processing. We also discuss an idea of how convolution works, convolutional neural network works, and the main operation types are used in building the convolutional neural network model. We can implement any other deep learning model withal but most advanced applications and models are currently being utilized by convolutional neural network so we decided to implement convolutional neural network for our practical implementation. After reading this section, you will have an early understanding of how deep neural networks work, and you will be able to move on to practical applications. Our goal is to give you an expeditious and facile tutorial on how to implement image relegation. Hopefully, you will be able to understand the main practical concepts and utilize this to build your applications and research.

A fundamental convolutional neural network model architecture Fig. 7 contains convolutional layer followed by an activation function (CONV), pooling layer (max or avg pooling based on the requisite) (POOL), dense/fully connected layer (FC).

3.3.1 Convolution Layer

To implement convolution operation kernels/filters are frequently utilized. The convolution operation (betokened by *) consists of multiplying the corresponding pixels with the kernel pixels, one pixel at a time, and summing up the values to assign that value to the central pixel. The same operation will then be applied, shifting the convolution matrix to the left until all possible pixels are visited. Kernels or filters are a matrix of values and the kernel slides over the input image and performs element-wise multiplication operation between the values in the filter with the pristine pixel values of the image. The multiplications are summed up engendering a single number for that particular receptive field. The input to the convolutional layer is an image that is resized to an optimal size (mundanely image size n*n) and fed as input to the convolutional layer. Let us consider image size is 32*32*1, where 32*32 is image dimension and 1 is the channel depth, it will be 1 for grayscale image and will take value 3 for color images.

3.3.2 Pooling Layer

Convolutional layers in a convolutional neural network methodically apply kernels/filters to images to extract optimized features and outputs feature maps. A quandary with feature maps is that they are sensitive to the location of the features in the input image which denotes a minuscule change (this can transpire due to shifting, cropping, rotation, or any other transformation) in the input image will yield different feature maps. A prevalent solution to this problem is downsampling. Mundane pooling methods are average pooling and max pooling. Max pooling is commonly utilized for the downsampling. Pooling layer operates on each feature map individually.

3.3.3 Dense/Fully Connected Layer

Fully connected layers or dense layers are a crucial layer of convolutional neural network, which are responsible for recognizing and classifying images or we can say that the final classification decision is taken by a fully connected layer. Fully connected layer takes the output from previous layers (convolutional and pooling layers of the defined convolutional neural network model) and predicts the class/label that best describes the input image.

3.3.4 Dataset

In this practical implementation, we will be working on one of the most extensively used datasets in malware classification that is the malimg dataset. The dataset details are given in Table 1. In this demonstration, we will build a simple convolutional neural network model to have an idea of the general structure of computations needed to tackle the multi-class classification problem.

First, let us understand the dataset. We are going to use malimg malware dataset [19] for practical purpose. Description of the dataset is as follows

1. The dataset contains 9339 malware images.
2. Malware images belong to 25 different malware families/classes.
3. Images are grayscale images.
4. All images of different sizes.
5. Dataset is highly imbalanced.

Table 1 Malimg dataset

Serial no.	Family/class	Family name	No. of variants
1	Worm	Allaple.A	2949
2	Worm	Allaple.L	1591
3	Worm	Yuner.A	800
4	Dialer	Instant access	431
5	Worm	VB.AT	408
6	Rogue	Fakerean	381
7	PWS	Lolyda.AA 1	213
8	Trojan	C2Lop.gen!G	200
9	Trojan	Alueron.gen!J	198
10	PWS	Lolyda.AA 2	184
11	Dialer	Dialplatform.B	177
12	Trojan-Downloader	Dontovo.A	162
13	PWS	Lolyda.AT	159
14	Backdoor	Rbot!gen	158
15	Trojan	C2Lop.P	146
16	Trojan-Downloader	Obfuscator.AD	142
17	Trojan	Malex.gen!J	136
18	Trojan-Downloader	Swizzor.gen!I	132
19	Trojan-Downloader	Swizzor.gen!E	128
20	PWS	Lolyda.AA 3	123
21	Dialer	Adialer.C	122
22	Backdoor	Agent.FYI	116
23	Worm:AutoIT	Autorun.K	106
24	Trojan-Downloader	Wintrim.BX	97
25	Trojan	Skintrim.N	80
—	—	Total	9339

3.3.5 Preprocessing

Our malimg dataset already contains malwarein the form of images (grayscale images), to demonstrate how a malware binary can be visualized as an image, we are going to use a random text file and we will show you how to convert the file into image Fig. 13. The ultimate goal of this step is to convert files into images and use them as the input of our convolutional neural network. We can convert any file using the following python code used by [19]. We have created a notepad file abc.txt with the contents of activeds.dll. Activeds.dll is the dynamic link library file of the windows operating system, which is stored in location c:/windows/system32/ activeds.dll.

The following python program is used to convert the abc.txt file into a grayscale image.

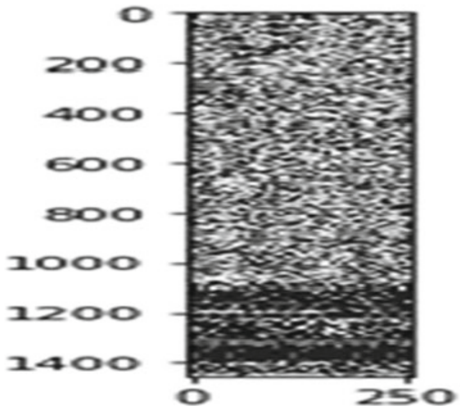


Fig. 13 Converted grayscale image

```
Python code to convert file into image
import numpy, os, array
import imageio
import matplotlib.pyplot as plt
from PIL import Image
filename = ’abc.txt’
f = open(filename,’rb’)
print(f)
ln = os.path.getsize(filename)
width = 256
rem = ln%width a = array.array("B")
a.fromfile(f,ln-rem)
f.close()
b=(len(a)/width,width)
b=numpy.uint(b)
g = numpy.reshape(a,b)
imageio.imwrite(’000f0e45f4f120838dd17500000de69a.png’,g)
plt.imshow(g,cmap=’gray’)
```

...

After running the above program we got the following grayscale image Fig. 13 of the corresponding abc.txt.

3.3.6 Image Resizing

Let us proceed, so our malimg dataset already contains malware samples in grayscale image format. To input these images into convolutional neural network for training, we need all images of the same size, so there is need to resizes all images of the malimg dataset to the specified size you want. I chose the (48*48) image dimension.

As we can see from Fig. 4, some differences among malware images, However, it would be too complex to accurately classify malware into their corresponding families as we have 9339 total malware images.

3.3.7 Implementation Details

For a programming perspective, we performed the following experiment utilizing a personal laptop with i3, 2.40 GHz intel processor; a 64-bit system with 4GB of random access memory. We used python programming language, python packages, and libraries which are availed to experiment. Keras library is utilized to train and test the model which utilizes a convolutional neural network. We have utilized spyder 4.0.1 which is a scientific python development environment tool, python 3.7.5 on windows 10, 64-bit windows 10 operating system.

3.3.8 Architecture

Let us define the convolutional neural network model for training and classification. Our dataset is ready; we have built our model using keras. Here, we will define our model, which is a stacked layer of convolution and pooling operations, with a final flattened layer and a softmax activation function applied to determine the class probability of the malware samples. The following network architecture will be used for training and testing purpose, the chosen convolutional neural network architecture Table 2 will only be for study and understanding purpose, We have randomly chosen the number of filters, layers, filter size. Hyper parameters tuning is also a research topic. So basically, we don't know how it is going to perform, what will be the accuracy, and we do not need to worry about these things here.

1. Convolutional layer : 30 filters, (3 * 3) kernel size, activation=ReLU
2. Max pooling layer : (2 * 2) pool size
3. Convolutional layer : 48 filters, (3 * 3) kernel size, activation=ReLU
4. Max pooling layer : (2 * 2) pool size
5. Dropout layer: dropping 50 percent of neurons
6. Flatten layer
7. Dense/fully connected layer : 1024 neurons, ReLU activation function
8. Dropout layer: dropping 50 perecnt of neurons
9. Dense/fully connected layer : number of output class, softmax activation function

Table 2 summarizes our chosen convolutional neural network architecture. model.summary() is used to visualize defined model architecture.

The input for convolutional neural network training has a shape of [48 * 48 * 1]: [image width * image height * channel /depth]. In our case, each malware is a

Table 2 Summary of our chosen convolutional neural network architecture

S. no.	Layer (type)	Output shape	Parameters
1	conv2d_3(Conv2D)	(None, 46, 46, 30)	300
2	max_pooling2d_3(MaxPooling2)	(None, 23, 23, 30)	0
3	conv2d_4(Conv2D)	(None, 21, 21, 48)	13008
4	max_pooling2d_4(MaxPooling2)	(None, 10, 10, 48)	0
5	dropout_3(Dropout)	(None, 10, 10, 48)	0
6	flatten_2(Flatten)	(None, 4800)	0
7	dense_3(Dense)	(None, 1024)	4916224
8	dropout_4(Dropout)	(None, 1024)	0
9	dense_4(Dense)	(None, 25)	25625

grayscale image, so the image channel value will be 1, if we use color images we have to assign value 3 in the image channel.We used the train test split() function of scikit learn to split dataset images between train and test, following a (90-10) % ratio.

Here is the code used to define convolutional neural network architecture using keras.

```
 #Python code to define convolutional neural network model architecture
model = Sequential ()
model.add (Conv2D (30, (3, 3), activation='relu', input_shape= (img_rows,
img_cols,img_channels)))
model.add (MaxPooling2D ((pool_size, pool_size)))
model.add (Conv2D (48, (3, 3), activation='relu'))
model.add (MaxPooling2D ((pool_size, pool_size)))
model.add (Dropout (0.5))
model.add (Flatten ())
model.add (Dense (1024, activation='relu'))
model.add (Dropout (0.5))
model.add (Dense (no_out_classes,activation='softmax'))
opti_mizer=Adam (lr=0.001)
model.compile (loss ='categorical_crossentropy', optimizer = opti_mizer,
metrics=['accuracy'])
```

We executed our program for 15 epochs. Epochs summary are as follows

```
Train on 7564 samples, validate on 841 samples
Epoch 1/15
7564/7564 [==============================] - 174s 23ms/step - loss:
```

```
0.9531 - acc: 0.7132 - val_loss: 0.2436 - val_acc: 0.9394
Epoch 2/15
7564/7564 [==============================] - 491s 65ms/step - loss:
0.2252 - acc: 0.9332 - val_loss: 0.1796 - val_acc: 0.9441
Epoch 3/15
7564/7564 [==============================] - 713s 94ms/step - loss:
0.1643 - acc: 0.9510 - val_loss: 0.1265 - val_acc: 0.9631
Epoch 4/15
7564/7564 [==============================] - 168s 22ms/step - loss:
0.1370 - acc: 0.9594 - val_loss: 0.1125 - val_acc: 0.9750
Epoch 5/15
7564/7564 [==============================] - 168s 22ms/step - loss:
0.1158 - acc: 0.9636 - val_loss: 0.1214 - val_acc: 0.9655
Epoch 6/15
7564/7564 [==============================] - 166s 22ms/step - loss:
0.1062 - acc: 0.9681 - val_loss: 0.0941 - val_acc: 0.9774
Epoch 7/15
7564/7564 [==============================] - 170s 22ms/step - loss:
0.0913 - acc: 0.9718 - val_loss: 0.1050 - val_acc: 0.9727
Epoch 8/15
7564/7564 [==============================] - 719s 95ms/step - loss:
0.0890 - acc: 0.9710 - val_loss: 0.1350 - val_acc: 0.9679
Epoch 9/15
7564/7564 [==============================] - 166s 22ms/step - loss:
0.0819 - acc: 0.9741 - val_loss: 0.0810 - val_acc: 0.9798
Epoch 10/15
7564/7564 [==============================] - 167s 22ms/step - loss:
0.0726 - acc: 0.9757 - val_loss: 0.1052 - val_acc: 0.9703
Epoch 11/15
7564/7564 [==============================] - 167s 22ms/step - loss:
0.0753 - acc: 0.9753 - val_loss: 0.0797 - val_acc: 0.9822
Epoch 12/15
7564/7564 [==============================] - 166s 22ms/step - loss:
0.0650 - acc: 0.9791 - val_loss: 0.1039 - val_acc: 0.9738
Epoch 13/15
7564/7564 [==============================] - 166s 22ms/step - loss:
0.0773 - acc: 0.9751 - val_loss: 0.0852 - val_acc: 0.9798
Epoch 14/15
7564/7564 [==============================] - 166s 22ms/step - loss:
0.0634 - acc: 0.9790 - val_loss: 0.0847 - val_acc: 0.9822
Epoch 15/15
7564/7564 [==============================] - 166s 22ms/step - loss:
0.0620 - acc: 0.9795 - val_loss: 0.0977 - val_acc: 0.9715
```

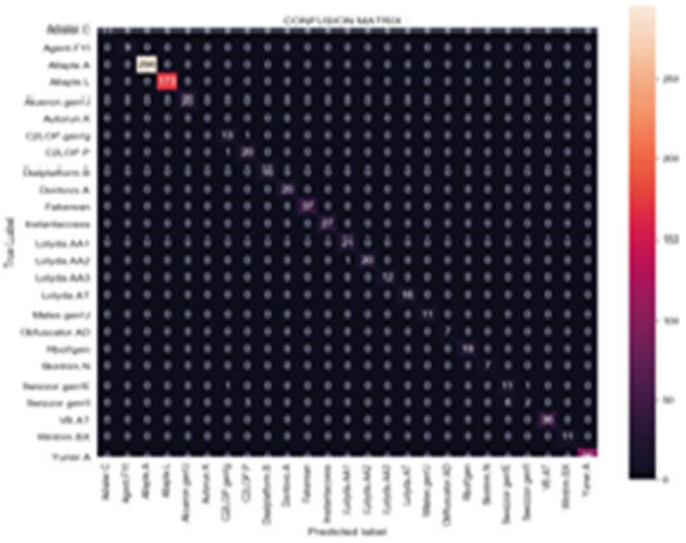


Fig. 14 Confusion matrix

3.3.9 Results

After training and testing our convolutional neural network model, we reached a final test accuracy of 97.537% which is very high! We got test time at 0.078. Here is the confusion matrix of our classification Fig. 14.

We can observe from the confusion matrix that most of the malware samples were well classified into its corresponding actual family, Autorun. K is always misclassified for Yuner. A, it is probably because we have only 80 samples of Autorun.K; this is very few in our dataset and that both are a component of a close worm type. Moreover, Swizzor.gen!E is often misclassified with Swizzor.gen!l, which can be explicated by the fact that they emanate from authentically close kind of families and types and thus could have homogeneous attributes in their code.We can also plot train and validation accuracy Fig. 15 and loss Fig. 16 during per epoch and analyze precision and losses, ups and downs during the whole journey. We can also calculate some more performance-based quantifications such as precision of the model which is 0.965, recall of the model is 0.975, and f1-score of the model is 0.968.

It is all about how to implement initial level malware image classification, and to further explore the results and analysis we can plot the confusion matrix, which give us some more statistics about the classification, some hints about what went erroneous during classification. It was the initial level understanding of how to implement

Fig. 15 Training and validation accuracy during 15 epochs

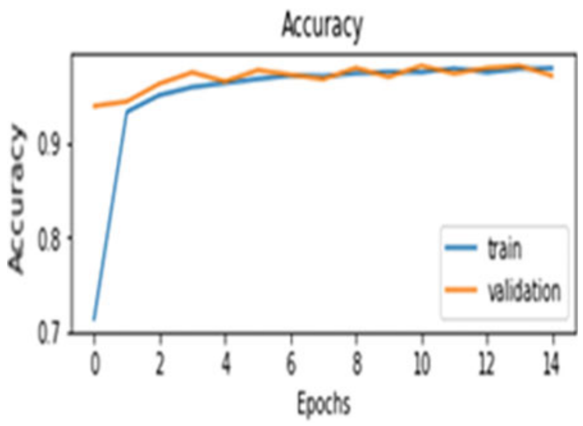
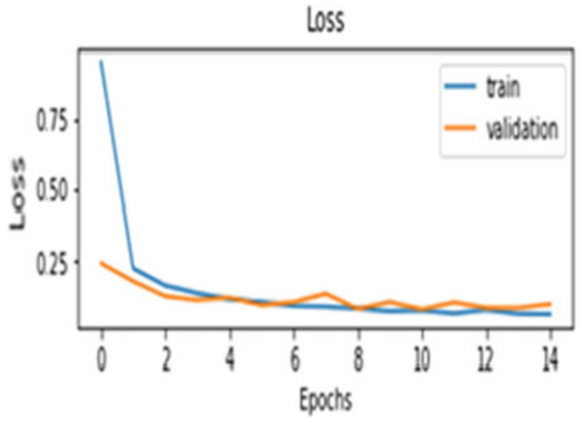


Fig. 16 Training and validation losses during 15 epochs



malware image classification utilizing convolutional neural network. Utilizing this basic understanding, you can further ameliorate classification results, perform more applications, and do further research.

3.3.10 Datasets for Malware Analysis

Here we are mentioning some popular datasets Table 3 available for practice and research. Datasets play a consequential role in training, testing, and validation of systems. Datasets of malware images consist of many images that belong to different families. Readers can make utilization of it for their research and projects. Some popular datasets are

Table 3 Summary of publicly available datasets for malware analysis

S. no.	Dataset	Dataset description
1	Malimg dataset	The malimg dataset consists of 9000 malware files belonging to 25 malware families and their variants
2	Malicia dataset	The dataset comprises 11,688 malware binaries collected from 500 drive-by download servers
3	Microsoft malware classification challenge dataset	This dataset contains 9 classes for training and testing purposes. It includes 21741 malware samples
4	Malshare	A free malware repository providing researchers access to samples, malicious feeds
5	IoT-23	A labeled dataset with malicious and benign IoT network traffic
6	AMD	Android malware dataset has 24,553 samples, it is integrated by 71 malware families ranging from 2010 to 2016
7	Android malware genome project	More than 1,200 malware samples that cover the majority of existing Android malware families, ranging from their debut in August 2010 to recent ones in October 2011
8	Drebin dataset	The dataset contains 5,560 applications from 179 different malware families. The samples have been collected from August 2010 to October 2012 and were made available to us by the mobile sandbox project

4 Challenges and Open Issues

This chapter and study is the first step toward enhancing our understanding of visualization and deep learning-based malware classification. During the study, many difficulties and challenges of malware classification were found; the present findings might have important implications for suggesting several courses of action to solve this problem. For a consistent and effective framework, it is important to address all the challenges and difficulties. Traditional malware classification approaches are very time-consuming and complex.

In our view, malware classification is very well handled by image visualization and deep learning approaches as compared to traditional approaches. Deep learning approaches efficiently perform learning but we found some limitations such as dl models requires all input images of the same size, which limits the training model. Work done by many researchers transform malicious binaries into grayscale images, so there is the scope of training and classification with color images. Different model architectures (differ in several layers, number of kernels/filters, size of strides, etc) show different results, so there is the scope of more intelligent architectures using deep learning to improve performance. Existing approaches which achieved high accuracy are often specific to a particular dataset, so still, there is a need for more generic deep learning architecture, which can be utilized for any type of dataset. The size of the dataset has significant importance on the accuracy and performance of the model. There is scope to address the data imbalance issue (as there is a difference in the number of samples in one family and rest). Available malware datasets constitute different formats and specifications containing both infected and non-infected files; inconsistencies may arise in the accuracy of the results, deep learning methods can be substantially influenced by adversarial attacks using the experience of the learning algorithm to avoid detection, or infuse harming instances into the training data. One of the latest emerging hurdles in malware analysis is the file less malware [10], which makes malware analysis more complicated. A combination of deep learning models for malware analysis can prove more intelligent and effective. To achieve good classification accuracy architecture alone is not only responsible it is also dependent on the dataset, so quality and enough data generate more accurate results, so preprocessing of the dataset is one of the important considerations for classification.

5 Conclusion

In this chapter, we provided a detailed study of the malware, malware analysis, deep learning, and its algorithms. The exponential development of the Internet, connected devices, services and applications, user's activity, and confidential information attracts cybercriminals. Although malware is not a new threat in the cyber world, but the device manufacturer, attacker's techniques to avoid detection and different service providers use different communication technologies creates a heterogeneous environment where malware analysis becomes a critical task. In this context, this chapter aimed to present an overview of the fundamental aspects of malware detection and classification using image visualization and deep learning techniques.

Within the next few years, malware classification and identification are likely to become important and inevitably be an issue that is going to be explored more. As can be concluded from the above-discussed information and study, the use of visualization techniques to represent malware and deep learning models in malware detection and classification proved to be efficient than traditional approaches. It is important to keep in mind that deep learning approaches prove to be a state-of-the-art approaches for malware detection and classification in some cases, but they are always the possibility of better to do. Deep learning methods also have some limitations such as a limited number of samples for analysis, to increase hidden layers which also increases complexity in the model and increases training time.

One biggest advantage of deep learning is automation. There is a need for automating the detection and classification process is still an important issue as most of the traditional malware analysis in the identification and classification of new threats continues to be a human task. Deep learning also has human interaction but limited sense. Until now, this methodology has only been applied to very less in literature and also in practice, so this chapter will encourage readers to do further research in this vast and important topic, and malware analysis is also connected to our lives directly. Malware classification is a fundamental and vital issue for future research and we have mentioned some state-of-the-art researcher's approaches, scope and emerging challenges for malware classification using deep learning for the reader's further studies. Finally, it is expected that the information presented and discussed in this chapter would help readers, analysts, and researchers to obtain a general and practical view of the malware analysis especially malware identification and classification from where they can visualize and explore new avenues of research.

References

1. Azab, A., M. Alazab, and M. Aiash. 2016. Machine learning based botnet identification traffic. In *2016 IEEE Trustcom/BigDataSE/ISPA*, 1788–1794.
2. Ding, Y., S. Chen, and J. Xu. 2016. Application of deep belief networks for opcode based malware detection. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 3901–3908.
3. Bagane Pooja, Garminla Sampath Kumar. 2020. Detection of malware using deep learning techniques. *International Journal of Scientific and Technology Research* 9: 1688–1691.
4. Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, ed. Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, vol. 27, 2672–2680. Curran Associates, Inc.
5. KyoungSoo Han, Jae Hyun Lim, and Eul Gyu Im. 2013. Malware analysis method using visualization of binary files. In *Proceedings of the 2013 Research in Adaptive and Convergent Systems, RACS '13*, 317–321. New York: Association for Computing Machinery.
6. Hardy, W., Lingwei Chen, Shifu Hou, Yanfang Ye, and X. Li. 2016. Dl 4 md : A deep learning framework for intelligent malware detection.
7. AV-TEST The Independent IT-Security Institute. Malware statistics and trends report [online] by av-test institute, 2020.
8. McAfee LLC is an American global computer security software company. Mcafee labs threats reports [online] by mcafee, 2019.
9. Jain, Mugdha, William Andreopoulos, and Mark Stamp. 2020. Convolutional neural networks and extreme learning machines for malware classification. *Journal of Computer Virology and Hacking Techniques*, vol. 04.
10. Sudhakar, K., and K. Sushil. 2019. An emerging threat fileless malware: a survey and research challenges 3: 1, 12.
11. Kalash, M., M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal. 2018. Malware classification with deep convolutional neural networks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 1–5.
12. Khan, R.U., X. Zhang, M. Alazab, and R. Kumar. 2019. An improved convolutional neural network model for intrusion detection in networks. In *2019 Cybersecurity and Cyberforensics Conference (CCC)*, 74–77.

13. Kim, C.H., E.K. Kabanga, and S. Kang. 2018. Classifying malware using convolutional gated neural network. In *2018 20th International Conference on Advanced Communication Technology (ICACT)*, 40–44.
14. Kim, Jin-Young, Seok-Jun Bu, and Sung-Bae Cho. 2017. Malware detection using deep transferred generative adversarial networks. In *Neural Information Processing*, ed. Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, 556–564. Cham: Springer International Publishing.
15. Lu, Y., and J. Li. 2019. Generative adversarial network for improving deep learning based malware classification. In *2019 Winter Simulation Conference (WSC)*, 584–593.
16. Mourtaji, Youness, Mohammed Bouhorma, and Daniyal Alghazzawi. 2019. *Intelligent Framework for Malware Detection with Convolutional Neural Network. NISS19*. New York: Association for Computing Machinery.
17. Naeem, Hamad. 2019. Detection of malicious activities in internet of things environment based on binary visualization and machine intelligence. *Wireless Personal Communications*, 1–21.
18. Naeem, Hamad, Farhan Ullah, Muhammad Rashid Naeem, Shehzad Khalid, Danish Vasan, Sohail Jabbar, and Saqib Saeed. 2020. Malware detection in industrial internet of things based on hybrid image visualization and deep learning model. *Ad Hoc Networks* 105: 102154.
19. Nataraj, L., S. Karthikeyan, G. Jacob, and B.S. Manjunath. 2011. Malware images: Visualization and automatic classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec '11*. New York: Association for Computing Machinery.
20. Ni, Sang, Quan Qian, and Rui Zhang. 2018. Malware identification using visualization images and deep learning. *Computers and Security* 77: 04.
21. Pal, K.K., and Sudeep, K.S. (2016). Preprocessing for image classification by convolutional neural networks. In *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 1778–1781.
22. Singh, Ajay, Anand Handa, Nitesh Kumar, and Sandeep Kumar Shukla. 2019. Malware classification using image representation. In *Cyber Security Cryptography and Machine Learning*, ed. Shlomi Dolev, Danny Hendler, Sachin Lodha, and Moti Yung, 75–92, Cham: Springer International Publishing.
23. Tobiyama, S., Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi. 2016. Malware detection with deep neural network using process behavior. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, 577–582.
24. Vasan, Danish, Mamoun Alazab, Sobia Wassan, Babak Safaei, and Qin Zheng. 2020. Image-based malware classification using ensemble of cnn architectures (imcec). *Computers and Security* 92: 101748, 05.
25. Yin, Qiwei, Ruixun Zhang, and XiuLi Shao. 2019. Cnn and rnn mixed model for image classification. *MATEC Web of Conferences*, 277: 02001, 01.
26. Yinka-Banjo, Chika, and Ogban-Asuquo Ugot. 2019. A review of generative adversarial networks and its application in cybersecurity. *Artificial Intelligence Review* 53: 06.