**An Artificial Intelligence Approach for Malware Detection using Deep Learning**

Saraswathi P1, Vamsi Krishna V2, Aidhitha K2, Venkata Yashwanth D2, Bindu M2

1 Assistant Professor, Department of CSE, GITAM School of Technology, GITAM Visakhapatnam, India. [spedada@gitam.edu](mailto:spedada@gitam.edu)

2Project Student, Department of CSE, GITAM School of Technology, GITAM Visakhapatnam, India.

**Abstract.** Computer users, organizations, and governments are all seeing an exponential development in the number of malware assaults. When it comes to determining the nature of unknown forms of malware, dynamic and static analysis of malware signatures and behavior patterns used by modern malware detection technologies are both inefficient and time intensive. The purpose of these investigations is to determine the types of malwares that are already known. Machine learning algorithms (MLAs) have been more popular as a method for successful malware analysis in recent years. This article investigates both conventional MLAs and deep learning algorithms or the detection using public and private datasets, classification, and categorization of malware. We provide a novel method to image processing, which is characterized by parameters that are optimal for the designs of deep learning conventional neural networks(DLCNN). A technique for the effective visual detection of malware is presented in its entirety in this study. The approach makes use of a framework for real-time deployments that is both scalable and hybrid in its fundamental makeup. The utilization of visualization and DL architectures for previous approaches(static, dynamic, and image processing-based hybrid) in an environment containing massive data is at the heart of a recently developed and significantly improved method for successfully detecting zero-day malware. In conclusion, the findings of the simulations revealed that the performance produced by the recommended DLCNN was better than the existing models. This was proved by the fact that the DLCNN outperformed the present models.

**Keywords:** Deep learning, robust intelligent malware detection, Machine learning algorithms (MLAs).

**1. Introduction**

The rapid advancement of technology has had an impact on both everyday operations of organizations and the everyday behaviors of people living in this Industry digital environment. IOT and applications are two technologies that have made important contributions toward the evolution of digital society in today's world. Concerns about safety, on the other hand, constitute a basic obstacle which is required to be overcome before the benefits of today's revolutionary industry can be completely realized. Cybercriminals target networks and individual personal computers in order to acquire confidential data in order to profit financially and to disable facilities. These kinds of attackers capitalize on malicious software, frequently referred to as malware, to severely compromise systems and take advantage of loopholes [1]. Software for computer systems built specifically to cause damage to the OS has been dubbed as viruses. A piece of malicious software can be referred to by a variety of names depending on its function and behavior like Adware, spyware, trojan horses, rootkits, backdoors, ransomware and command. Each of these names refers to a specific type of malicious software. The detection and prevention of harmful softwares are ongoing challenges in the area of information technology security. Researchers' creation of new techniques have allowed malware authors to further enhance their abilities to elude identification.

Following the Morris worm, a groundbreaking computer virus, arose in 1988–1989, anti-virus software programmes evolved to detect the malware presence by striving inorder to find an association with a regularly updated database of virus definitions. This permitted the software to stay current with the ever-changing threat landscape. This enabled the ability for recognising fresh viruses as they evolved. The term "signature-based malware detection" applies to this approach to detect notorious malware, in order to identify the features of malware's activities, it could also perform an intuitive search. However, the primary difficulty with those traditional methods is that modern malware uses antivirus avoidance strategies like code obfuscation, making it infeasible for these signature-based techniques to effectively identify zero-day malware. [2].This is a significant challenge. Signature-based malware detection systems need a large amount of domain-level knowledge since they use different kinds of analysis like static analysis and dynamic analysis to be competent to allocate a sign to a part of malicious software. In addition, it consumes a longer duration and period of time to develop malware that could be replicated on a system based on signatures, and during that time, an adversary may be able to get into the system. Systems that rely on signatures fail to recognise newly emerging malicious application attacks, which is another drawback of using this approach. Polymorphism and metamorphism are the two most prevalent strategies that hackers use in order to avoid being detected by signature-based systems, as established by the research conducted by specialists in the area of information security. To eliminate this problem, application programming interface, or API, calls are manually examined through programmes that were originally manually gathered with software tools. To tackle the issue, this is done.

[3] offered a system to automatically extract API requests and conduct an analysis of the potentially damaging features utilizing a four-step methodology.This technique is a labor-intensive endeavor that necessitates a lot of resources, thus [3] constructed the system. Detection of the malware-ridden programme is the first step. Step 2 involves disassembling the binary executable into its component pieces. The following phase involves doing the extraction of API calls. The fourth phase involves mapping API calls and performing statistical feature analysis. This was enhanced in [4] by employing a five-step approach that employed a machine learning algorithm (MLA) like Support Vector Machine with n-gram features gathered via utilizing tenfold cross validations, large samples of executables that are both benign and dangerous so this was done. In a subsequent paper [5], an evaluation and comparison of a large number of classical ML classifiers for detecting malware. In addition, throughout the course of this study, a framework for the identification of zero-day malware was developed. In order to control potentially harmful code variants, the order in which API calls were made as well as the frequency with which API queries were made were input into methods for similarity-based mining and machine learning [7]. A rigorous qualitative study was carried out on an extensive information set, and a unifying structure was established. In order to extract attributes from malware binaries. Unique feature selection approaches for superior surveillance of malware were demonstrated utilizing the study referred to as [8]. Support vector machine (SVM) and Maximum-Relevance Minimum Redundancy Filter (MRMRF) heuristics combined with API queries to create features. Recently, in response to a rise in assaults carried out by unknown malware, in-depth [6] has provided information on obfuscated malware, and [9] and other researchers are trying to enhance MLAs for malware detection. This serves as the impetus for the conduct of this investigation.

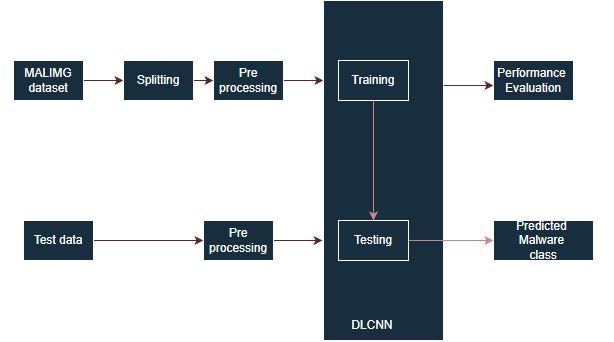
**2. Background and Related Works**

The use of the techniques of characteristic engineering, characteristic selection, and characteristic representation are fundamental to the operation of MLAs. In order to train the model, a set of attributes are used, which each corresponds to a distinct division. This training's goal is to draw a distinction between software that is safe and software that is detrimental. The detection of malware and the subsequent categorization of that malware into the family to which it belongs are both aided by this separation plane. Learning the underlying domain is required for proper functioning of both the engineering of features and the methods used for choosing features. Employing both dynamic and static analysis, it is attainable to gain an abundance of attributes. By using an algorithm called "static analysis," which does not run a binary programme through its motions in the traditional sense, information may be derived from a binary programming. Dynamic analysis is a manner of retaining a close watch on malicious software's movements while it executes in a setting independent of the rest of the system. (C. Rossow et al., 2012) explores the challenges in greater detail of dynamic analysis and handles issues that research raises. Using dynamic data analysis is projected to be a long-term profitable and successful approach of recognising malware. Since it takes several seconds to assess the object's behavior under examination, dynamic analysis is not suitable for use in endings that detect malware. This is precisely a potentially destructive payload that could be broadcast while analysis of its habits is taking place. Because techniques are more impervious to this kind of manipulation, dynamic analysis-based malware detection techniques are less liable to be tricked by techniques for obfuscation. This is due to the fact that, in comparison to statically obtained data, it is more difficult to alter. Both analysis techniques are combined by the vast bulk of commercial anti-malware programmes. This is the case for both locating the problem and fixing it. The fact that the traditional ML based detection systems depend on approaches for feature engineering, feature learning, and feature representation, all of which need a high order of expertise in the area. [11], [12], and [13]—is the primary thing with these types of computers. Furthermore, Suppose the attacker learns the features, escaping malware detectors is effortless (Y. LeCun et al. 2017). This is a consequence of the malware detector hunting for those characteristics. For MLAs to be productive, the data needs to include several kinds of malware patterns. The quantity of comparative data accessible to everyone else for research is accurately little due to considerations about privacy and security.Despite the limited number of datasets, many are obsolete, therefore nearly each of them has substantial flaws. The authors own datasets have been used extensively in previously published ML based malware analysis results. Even though there are sources that are openly accessible to crawl the malware databases, creating the ultimate dataset for research reasons is difficult. Creating a generic malware analysis system that is based on ML and to be deployed in actual time presents many obstacles; the most important of these issues are detailed further down in this section. Even more importantly, the compelling problems that come with In-depth discussion of using data science tools was written in (R. Verma et al. 2018).

This is the case in many of the tasks that exist in computer vision fields, speech processing, and many other fields. This is the case with many of the jobs that are performed in the disciplines of computer vision, voice processing, and a great deal of other fields as well. This is the case due to the fact that deep learning is essentially an improved version of the neural network model, which explains why this is the situation. It aspires to obtain a more complex representation of the traits found in the deeper layers beneath as part of its training, and it is able to utilize lessons from its past errors. The performance of tasks may be improved via the use of deep learning, which involves the collection of new patterns and the establishment of connections between previously gathered patterns. ML Algorithms acquire outputs which continue diminishing as they continuously view more data, but deep learning accumulates innovative patterns and increases performance as more data is viewed. MLAs have reported witnessing outcomes that are becoming worse as they continue to take in more and more data. There have only been a few research articles [13], [11], [12], [17], [18], [18–24] carried out on the application regarding DL architectures to malware research. These studies are cited as follows: [13], [11], [12], [17], [18], [18–24]. Despite this, there has been a worrying rise in the number of malicious software applications over the course of the last several years as a direct effect of Industry 4.0. Big Data is produced as a consequence of the continuous collecting of malwares in real time, which is an indication that the currently available technologies are not scalable and have very high demands for the amount of storage space and time necessary to make correct judgments.

**3. Proposed Methodology**

Artificial intelligence's deep learning discipline pulls its inspiration from the method in which the brain performs out its numerous tasks. Deep learning tends to be referred to as neural networks with deep layers, or DNNs. Two biggest advantages deep learning systems have are the capacity to spontaneously change meaning derived from being able to comprehend the meaning of data despite the volume of information to be analyzed also capacity while processing new data without seeking domain knowledge..Real-world applications often make use of a variety of deep learning designs, alike recurrent neural networks (RNNs) and conventional neural networks (CNNs). Both of these types of architecture are manifestations of distinct sorts of deep learning. RNN structures will be typically used for temporal data, whereas CNN patterns are typically used for geographic data. Neural networks are capable of processing both kinds of data.Using neural networks, one can handle any form of the aforementioned data. When used in conjunction with one another, CNN and LSTM are able to do analysis on data that is both spatial and temporal in nature.



**Fig. 1**. Proposed block diagram.

The suggested methodology is shown as a block diagram in Figure 1.The MALIMG dataset is initially divided into 80% training data and 20% testin data. After that, a dataset preprocessing procedure is carried out in order to standardize the whole dataset. In addition, the DLCNN classifier is used for predicting malware attacks based on given sample data. The purpose of the performance assessment demonstrates the superiority in this approach.

**3.1 MALIMG dataset**

CICDoS2019 includes DDoS attacks that are not dangerous and use the greatest and newest versions; these attacks mimic the real-world data (PCAPs).In addition to this, it takes into account the findings of the network traffic analysis that was carried out with CICFlowMeter-V3 and includes labeled flows that are arranged in a manner that is consistent with the time stamp, protocols, attack type, source and target IP addresses, source and destination ports. Moreover, it contains a labeled flow that was labeled with an attack type (CSV files). When we were constructing this dataset, one of our primary goals was to create realistic background traffic as one of our core objectives.In the testbed that we have proposed, our B-Profile technology has been a method for describing the abstract behaviour of human encounters. In addition to that, our system generates realistic non-intrusive background traffic. For this dataset, abstract behavior of 25 users based on email, HTTP, HTTPS, FTP, and SSH protocols.

**3.2 Pre-processing**

Cleaning and prepping raw data so that a machine learning model can use it is referred to as "data pre-processing."This stage has to be finished before the data can be used in any way. This phase, which is both the first and most important step, kicks off the procedure to generate a model for ML and is the process's starting point. When we are utilizing machine learning to design a project, we may not always come across data that is both clean and organized. This might be challenging for us. This is due to the fact that it does not always hold true. In addition, before carrying out any action that includes data, it is required to thoroughly clean the data and arrange it in the right manner. This is a prerequisite for carrying out any activity that involves data. Before moving on to any other steps that include data, this one has to be completed first. In light of this, the activity of pre-processing the data is used.

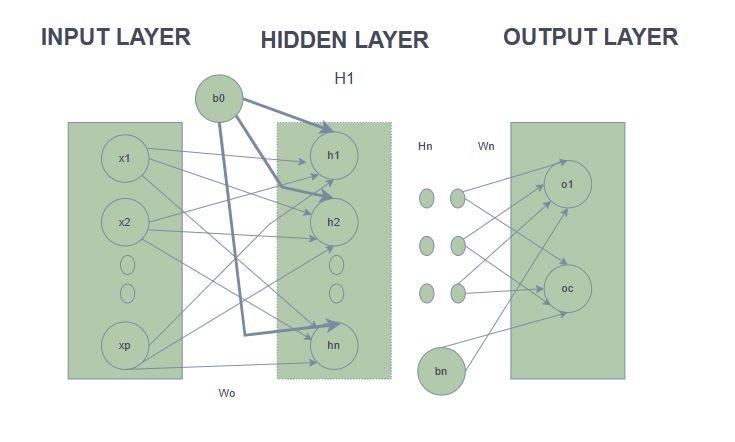
**3.3 Splitting the Dataset**

During the pre-processing phase of machine learning, our dataset is split into the training set or the test set, into two distinct categories. These categories, accordingly, correlate to the training set and the test collection. When it comes to the data pre-processing stage, this is one of the most important processes because it allows us to improve the overall performance of our ML model. As a result, it is among the most crucial processes. Think about the following example: we trained our ML model on one dataset and then tried it on a unique dataset. If this happens, our model will have a harder time comprehending the connections between the different models .If the model is adequately trained and consists of greater accuracy, but when a new dataset is given, Its efficiency will decline; however, if we train it inefficiently, it will perform better. There will be a performance slowdown if our model is very well trained and has high training accuracy.

**3.4 DLCNN**

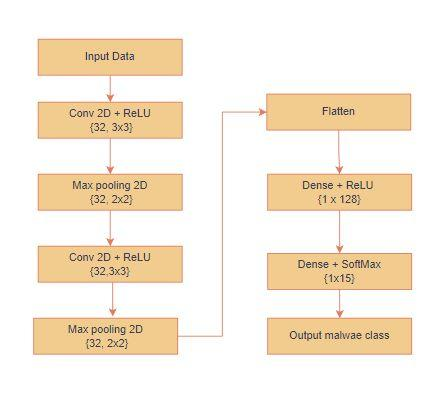
The generation of a directed graph is accomplished via a feed forward neural network, commonly referred to as an FFN. According to the concept of graph theory, the components of a graph are nodes and edges. Without the construction of a cycle, the data in an FFN network is sent along the edges from one node to the next as it travels from network to network. A (MLP) structure, which is an extension of FFN, has three or more levels: an input layer, one or more secret layers, and an output layer. The words input, hidden, and output layers, respectively, serve to signify each of these layers. The precise elements that make up a layer stack are an input layer, one or more hidden layers, and an output layer. (MLP). In the notation used for mathematics, the term "unit" refers to the large number of neurons that can be found in each of the layers that make up this structure. Hyper parameter tuning is utilized in calculating overall hidden layers. They are implementation. Although the data is translated from a single layer to next in front direction, the data of the layers that passed before it are disregarded in the process. In addition, each layer's neurons are connected to their neighbors and to the neurons in the layer below.

The primary use for a CNN is in the area of data processing. A CNN is a complement to the traditional feed forward network (FFN) shown in Figure 2, but all the connections, hidden levels, or its units are not displayed in that diagram. In this context, m refers to the no.of filters, ln to the number of input features, and p to the decreased feature dimension; the latter variable's value is determined by the pooling length. The convolution 2D layer, the pooling 2D layer, and the layer with complete connectivity are the three layers that collectively make up the CNN network that was used in this study. In a CNN structure, several completely interconnected layers, convolutional 2D layers, and pooling 2D layers are all feasible. The finest characteristics are extracted when the sequence data from the 1D layer is passed through the filters in the recurrent 2D layer. A new feature collection called the feature map is created by extracting features from each filter and combining them. A hyper parameter tuning technique is used to precisely select the length and total number of filters. This then uses ReLU, a non-linear activation function, to independently activate each component.



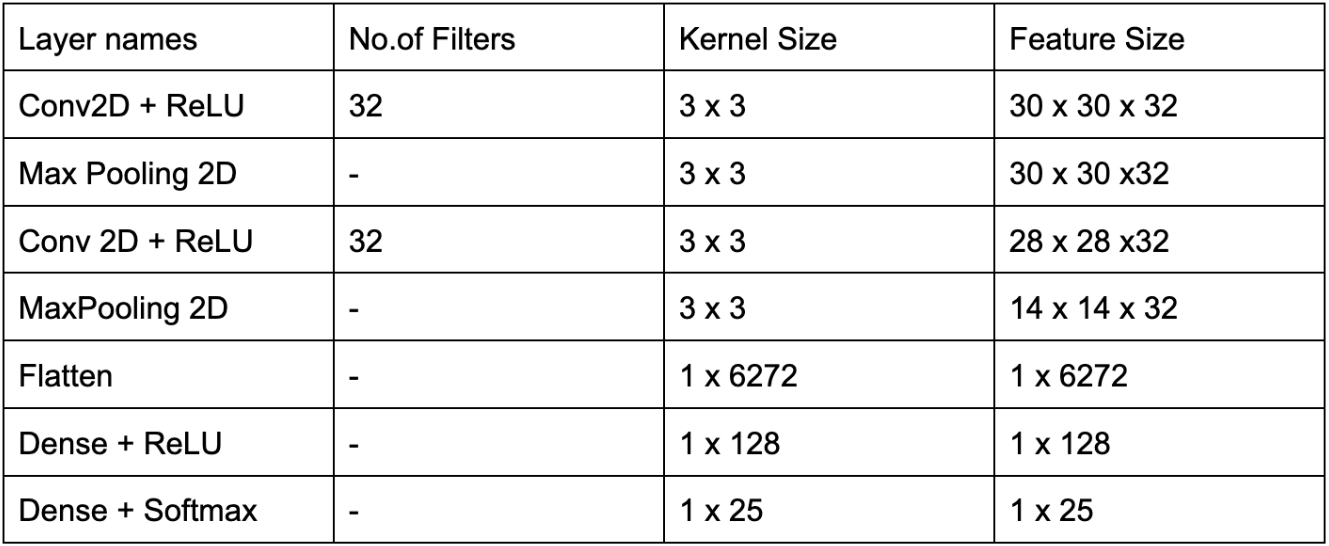
**Fig. 2**. DNN architecture.

By using either of the pooling techniques, the dimensions of the ideal features may be reduced in size. This can be done using either the max pooling or the min pooling. Max pooling is utilized for this investigation's objectives because it enables us to pick the highest achievable output in the confines of one particular spot. In conclusion, the DLCNN network has a fully connected layer that may be submitted for classification. A layer of neurons that are completely linked are related to every other neuron in that layer. This indicates that there is communication between all parts of the layer. In addition to sending the advantages of the pooling 2D layer into a completely connected layer, sequence-related data can also be obtained by sending the features of the pooling 1D layer to LSTM. This can be done in order to achieve the same goal as the fully connected layer. In the last phase of the process, the LSTM features are integrated into the fully linked layer so that they may be classified.As a result, the characteristics can be divided into various categories.

****

**Fig. 3**. Architecture of DLCNN.

**Table 1**. Layers description.



Conventional neural networks are, for the most part, built up of three distinct parts, each of which makes up the overwhelming bulk of the network. Convolutional layer, which is used for the purpose of differentiating characteristics from one another. The convergence layer, which is also sometimes referred to as the pooling layer, is the one that is used for feature selection the majority of the time. It is possible to reduce the overall number of parameters if the number of characteristics that are being used is reduced. It is the responsibility of the whole connection layer to carry out this task, and it is also the function of this layer to provide a summary of the features. One component of a convolution layer is a convolution process, while the other component is a nonlinear activation function known as ReLU. Together, these two parts make up the convolution layer. The construction of a typical CNN model, as seen in Figure 3, is shown here for the purpose of identifying various categories of malicious software.

The data that is located on the layer to the left is the input layer, and the machine interprets this layer as the input for a number of different matrices. Convolution layer is followed, which employs ReLU as its activation function. This layer comes after the previous one. Within the pooling layer, there is not an activation function that can be found. The mixture of the convolution and pooling layers may be produced in a few different ways. These possibilities are listed below. During the process of constructing the model, the various combinations of convolution layers and convolution layers, as well as convolution layers and pool layers, may be very freely mixed and are not constrained in any way. Convolution layers and pool layers may be combined to form a single layer. On the other hand, the most frequent sort of CNN is a mixture of a large number of distinct layers of convolution and pooling. This kind of CNN is called a multilayer convolutional network. The freshly acquired feature representation is transported to the sample label space by a finished connection layer, which also serves as a classifier. Although it is the last, this is not the smallest move.

In addition, the weight parameter that is associated with each connection is completely unique to that link. The size of the parameters becomes proportionally larger as the number of layers increases. On the one hand, this will make it much simpler for the over-fitting phenomena to manifest itself. Neural being too complex and might result in a significantly reduced level of success throughout the training. Each individual piece that makes up the conventional kernel will perform an operation on a specific point inside each local input. The approach of parameter sharing, which is used in traditional neural networks, is what makes this possibility a reality. To operate properly, a neural network must be trained with a specific set of parameters. It is not necessary that the learning be maximized for each particular parameter.

The local invariant feature of the data shows that the original data will not be changed even if the data size is scaled, translated, or rotated. This characteristic is referred to as the stability of the data, and it is referred to as the local invariant feature. Because in DL, improving the data is necessary in raising the accuracy, preserving the local invariance of the data while using a fully connected feedforward neural network may be a challenge. In order to overcome this obstacle, the conventional operation in a conventional neural network will need to be carried out.

**DLCNN Layers**:A kernel or filter, rectified linear unit (ReLU), max pooling, fully connected layer and SoftMax layer with the layer of classification are some of the layers of convolution that are used in DLCNN training and testing to allow each source data to classify objects with probabilistic values between [0,1].

Fig 4 demonstrates that the primary layer that is used to extract features from a source data is called the convolution layer. This layer also preserves the link between pixels by learning the characteristics of data by making use of very small blocks of the source data. This layer might be considered the main layer since it is responsible for extracting features from a data source. It is a mathematical function that takes into account two inputs, such as the source data and the spatial coordinates, which are indicated by the number of rows and columns in the table. These inputs are taken into consideration in the function. The notation may be used to indicate a filter or kernel that has the same size as the input data if it has the same dimension as the data being represented, which in this example is RGB. The notation can also convey the dimension of the data .

Shape

Description automatically generated

**Fig.4**. Convolution layer process representation.

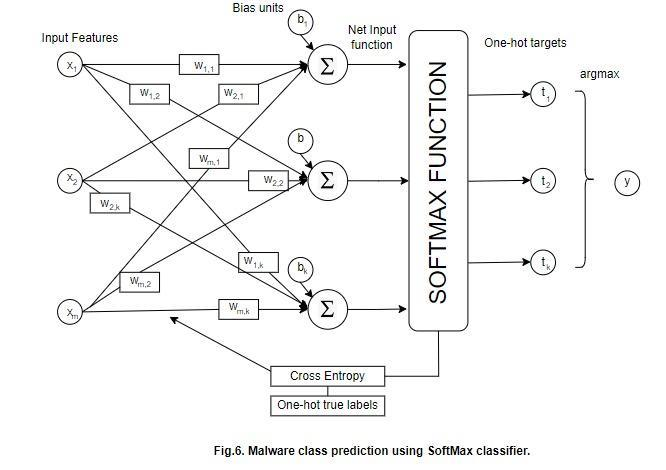
The size of the output that was generated via the convolution method of the input data and the filter was known as feature map. Figure 5 provides you with an illustration of one possible method of convolution for your review. This was done for your convenience (a). Imagine for a moment that the size of the filter is, and that the size of the data that is being fed is. Let's suppose for the sake of this discussion that the data being produced by the output is. It is possible to acquire the feature map of the input data by performing the operation indicated in Figure 5, which involves multiplying the values of the input data with the values of the filter. The steps involved are presented in Fig 5. (b).

| A picture containing text, crossword puzzle  Description automatically generated  5.(a) | A picture containing diagram  Description automatically generated  5.(b) |
| --- | --- |

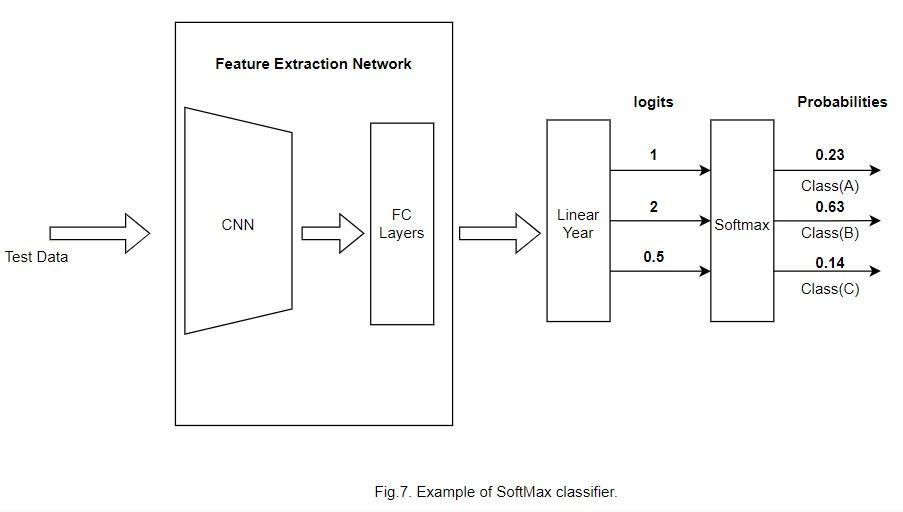
**Fig. 5**. Convolutional layer processing example (a) a data set of size is convolving with kernel (b) Convolved feature map

SoftMax function is often used in the very last step of the output process, as the picture to the right illustrates for your viewing pleasure. This is because the nodes will ultimately meet at the output's end, and as a result, they will be in a position where they can be classified. This is the reason why this is the case. Here, X serves as the input for each of the models; the layers that are situated in the space between X and Y are the ones that are considered to be the hidden layers; data travels from X to Y, passing through each of the layers along the way; and Y is the output of the whole procedure. Let's pretend there are ten distinct classes, and that our mission is to figure out which of those classes the information that was provided belongs in. As a consequence of this, the step that we do is to designate a certain output expectation for each class. This means that we have ten outputs, each of which fits into a different class, and we forecast the class based on which class it is most likely to belong to. In the case of Figure 6, we are tasked with making an educated judgment as to the type of object that can be seen in the picture. In most cases, we are in a position to establish whether or not the virus in question is type A.

However, given the circumstances of this specific situation, we will have to hazard a guess as to the nature of the object that can be seen in the shot. At this juncture, the usefulness of softmax truly starts to come into its own. mostly as a result of the fact that the model was based on some data at some point in the past. As a consequence of this, as soon as the image is delivered, the model immediately does an analysis of the images, then sends it to softmax with the intention of classifying the picture. One-hot encoding is a strategy that is used by the softmax algorithm in order to quantify the amount of cross-entropy loss and establish what the maximum value is. One-Hot Encoding is the name of the procedure that is used throughout the process of the data being categorized as it should be. In the example that came before, the One-Hot Encoding would have been applied to the following in the event that softmax had anticipated that the item was a member of class A:



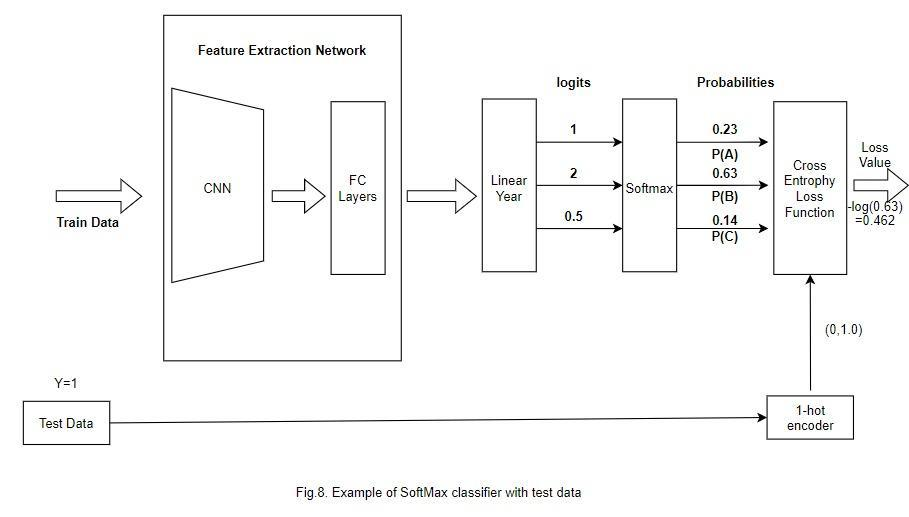
**Fig.6**. Malware class prediction using SoftMax classifier.



**Fig.7**. Example of SoftMax classifier.

Class (A) will be [1 0 0], Class (B) will be [0 1 0], Class (C) will be [0 0 1]

From Figure 7, It's possible that our predictions have been proven accurate at this point. However, we are unable to make correct predictions the vast majority of the time. On the other hand, the algorithm is charged with picking the predicted item that is the most correct. A computer employs a function known as the cross-entropy function in order to correctly identify an object. This is necessary for the computer to do so. The following approach to computing cross-entropy gives us the opportunity to choose values that are more directly similar to those of other variables.



**Fig.8.** Example of SoftMax classifier with test data.

In the above Figure 8, When we take a look at function loss, we find that the value is 0.462. We find loss for the remaining classifiers by using the same methods. In order to provide the most accurate forecast feasible, the loss function ought to be as small as is humanly possible. The list of potential mathematical representations of loss function that follows is: -

**4. Results and Discussions**

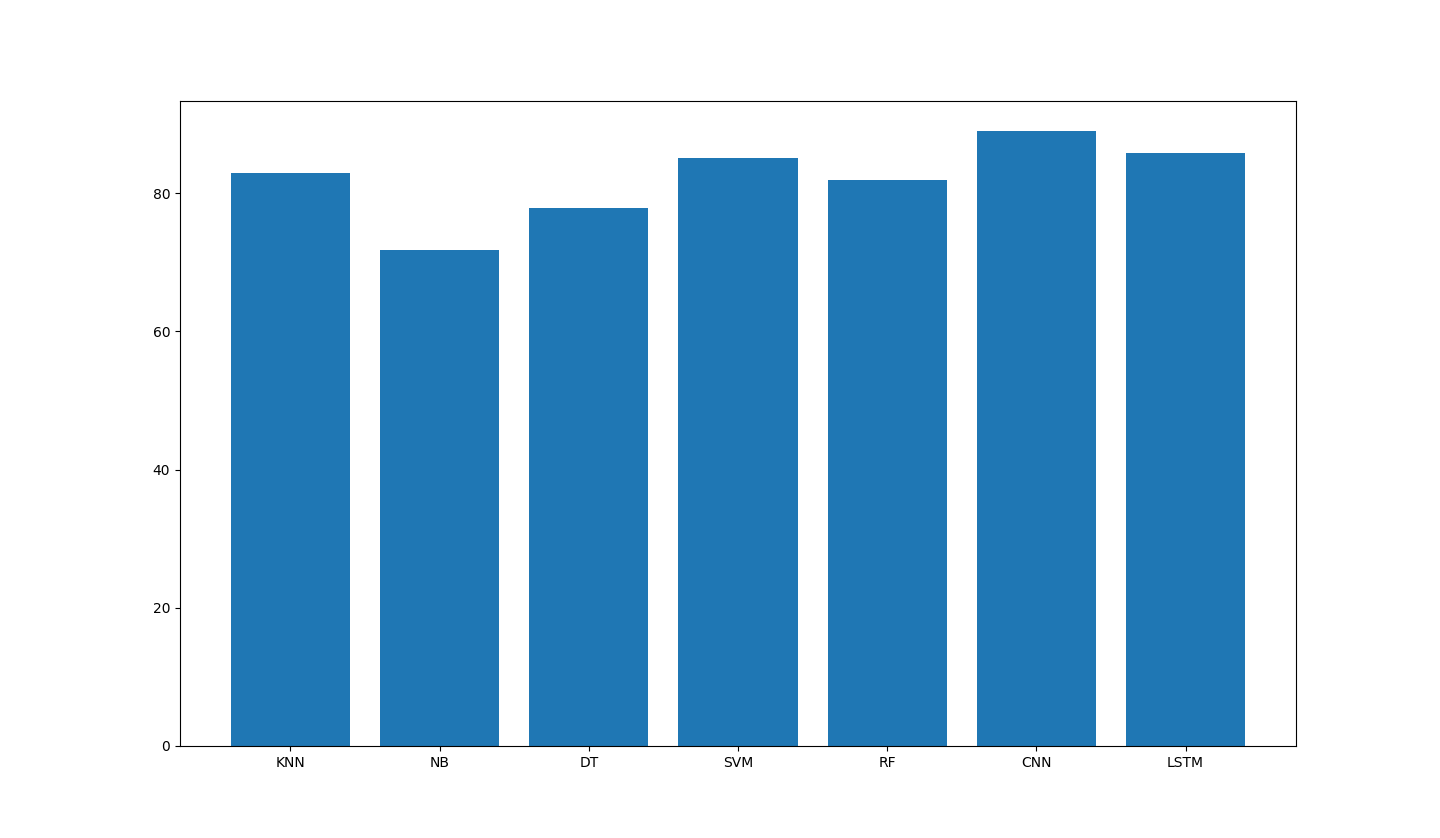
Before being used in the production of the dataset, malicious binary files are first transformed into a binary matrix format. This matrix is represented as an unsigned integer with eight bits of data. 0 denotes lackness and 255 denotes brightness. When seen in this manner, this matrix is much simpler to comprehend compared to its original form. Size of an array that formed after transforming the two-dimensional matrix into the one-dimensional vector form was 1x1024. L2 normalization is utilized whenever newly created data are analyzed for any purpose.The dataset was subsequently separated via randomization into two independent sets: 80% for training and 20% for testing.

We have used the following preprocessing processes in order to get the datasets ready for the experimental study that will be carried out:

1) Ember: Different characteristics are retrieved from a parsed PE file using domain-level knowledge. These traits include features that are independent of format, such as the raw byte histogram and the byte entropy histogram. Following the acquisition of these properties from [27], strings are then introduced into the LightGBM model. They use LightGBM's default parameters, which consist of a gradient boosted decision tree (GBDT) with 100 trees and 31 leaves per tree. These values were chosen at random. This is because the LightGBM model performs better than the MalConv model does, which is the reason for this result. The explanation for this may be found in the simple fact that. After this, we will continue to analyze the effectiveness of traditional MLAs and DNNs for the classification of malware by utilizing the Ember dataset as our primary resource.

2) MalConv: MalConv architecture was published in [11] with the objective of identifying malware. MalConv was developed for this reason. It is made up of three distinct components, each of which is subjected to the following procedures: Pre-processing, convolution, and full linkage. This creates it. The embedding layer receives the extracted binary file byte sequences during pre-processing. This step takes place before the actual processing begins. Due to the fact that each embedding has a dimension of 8, the overall size of the dictionary of embeddings is 257, which corresponds to the total size of the embedding layer, which is 257.The embedding layer is responsible for converting bytes into representation of vectors that has been specified. You'll discover two 1D layers of convolution if you look at the convolution section of the MalConv program. Each one-dimensional conventional layer consists of 512 units (with a kernel size of 4 and 128 filters), in addition to 500 strides. When it comes to these layers of the conventional filter, the gated convolution approach is the one that is applied. The pooling length in the convolution layer is 4,000, and the technique that it uses is called a temporal max pooling algorithm. Aside from contributing in reducing dimension, this also resolves another issue of information sparsity. Two layers of completely linked sections serve to establish a completely connected section. There are 128 nodes in the first fully connected layer, and only one node with a sigmoid non-linear activation function in the fully connected second layer. Both layers are fully linked. A combination of LSTM and SVM is applied just before the very last step of the classification process.

3) The MalConv Model Has Been Interpreted in Several Ways MalConv Model with SELU nonlinear activation function and DeConv regularization removed [12]. Strides somewhat altered. There are four total convolution layers in the convolution section. Specifically, there are four convolution layers followed by two convolution layers, a max pooling step, two more convolution layers, and finally two more convolution layers. There are 32 units across the first couple convolution layers with 4x strides, and 16 units across the next two layers with 8x strides. Both of the final layers, which were constructed via global average pooling, are intrinsically linked to the earlier ones (the first four levels).



**Fig. 9**. Performance comparison of accuracy obtained using existing and proposed models.

**5. Conclusions**

In this study, we built a highly scalable framework for the malware detection, classification, and categorization of zero-day malware. Additionally, we provided an effective method for the detection of malware and identified an efficient strategy for the detection of malware. Both of these contributions were made in the context of the study. This framework employs a two-stage process to carry out malware analysis and makes use of DLCNN to do the analysis of malware that have been obtained from the hosts of end users. In the initial stage of the process of classifying malware, a mix of static and dynamic analysis was used, as described in the previous sentence. During the second stage of the procedure, a number of different methods of image processing were used in order to categorize various forms of malware into their respective groupings. Applying various models to benchmark datasets that were available to the public allowed for the successful completion of a variety of different experimental investigations. According to the findings of these studies, the model that was developed was much better than the conventional MLAs. The framework that has been proposed is authorized to perform analysis on significant different types of malwares. In addition, through the addition of a few additional layers to the present structures, the framework could possibly be enhanced for evaluating a still broader spectrum of malware varieties. The investigation of these variations need to be included in future research, along with the inclusion of new components with present data.

**6. References**

[1] R. Anderson et al., ‘‘Measuring the cost of cybercrime,’ in The Economics of Information Security and Privacy. Berlin, Germany: Springer, 2013, pp. 265–300.

[2] B. Li, K. Roundy, C. Gates, and Y. Vorobeychik, ‘‘Large-scale identification of malicious singleton files,’’ in Proc. 7th ACM Conf. Data Appl. Secur. Privacy. New York, NY, USA: ACM, Mar. 2017, pp. 227–238.

[3] M. Alazab, S. Venkataraman, and P. Watters, ‘‘Towards understanding malware behaviour by the extraction of API calls,’’ in Proc. 2nd Cybercrime Trustworthy Comput. Workshop, Jul. 2010, pp. 52–59.

[4] Rao, P. M., & Deebak, B. D. (2022). Security and privacy issues in smart cities/industries: Technologies, applications, and challenges. Journal of Ambient Intelligence and Humanized Computing, 1-37.

[5] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, ‘‘Zero-day malware detection based on supervised learning algorithms of API call signatures,’’ in Proc. 9th Australas. Data Mining Conf., vol. 121. Ballarat, Australia: Australian Computer Society, Dec. 2011, pp. 171–182.

[6] Deepak, S., Kumar, S. U., & Durgesh, M. (2012, May). Query processing and optimization of parallel database system in multi processor environments. In 2012 Sixth Asia Modelling Symposium (pp. 191-194). IEEE.

[7] M. Alazab, ‘‘Profiling and classifying the behavior of malicious codes,’’ J. Syst. Softw., vol. 100, pp. 91–102, Feb. 2015.

[8] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, ‘‘Hybrids of support vector machine wrapper and filter based framework for malware detection,’’ Future Gener. Comput. Syst., vol. 55, pp. 376–390, Feb. 2016.

[9] E. Raff, J. Sylvester, and C. Nicholas, ‘‘Learning the PE header, malware detection with minimal domain knowledge,’’ in Proc. 10th ACM Workshop Artif. Intell. Secur. New York, NY, USA: ACM, Nov. 2017, pp. 121–132.

[10] C. Rossow, et al., ‘‘Prudent practices for designing malware experiments: Status quo and outlook,’’ in Proc. IEEE Symp. Secur. Privacy (SP), Mar. 2012, pp. 65–79.

[11] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas. (2017). ‘‘Malware detection by eating a whole exe.’’ [Online]. Available: <https://arxiv.org/abs/1710.09435>

[12] M. Krcál, O. Švec, M. Bálek, and O. Jašek. (2018). Deep Convolutional Malware Classifiers Can Learn from Raw Executables and Labels Only. [Online]. Available: <https://openreview.net/forum?id=HkHrmM1PM>

[13] M. Rhode, P. Burnap, and K. Jones, ‘‘Early-stage malware prediction using recurrent neural networks,’’ Comput. Secur., vol. 77, pp. 578–594, Aug. 2018.

[14] H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, Evading Machine Learning malware Detection. New York, NY, USA: Black Hat, 2017.

[15] Patruni, M. R., & Saraswathi, P. (2022). Securing Internet of Things devices by enabling Ethereum blockchain using smart contracts. Building Services Engineering Research and Technology, 43(4), 473-484.

[16] Sivaranjani, R., Rao, P. M., & Saraswathi, P. (2021). ECC-Based Privacy-Preserving Mechanisms Using Deep Learning for Industrial IoT: A State-of-the-Art Approaches. Deep Learning for Internet of Things Infrastructure, 25-59.

[17] Rao, P. M., & Deebak, B. D. (2023). A Comprehensive Survey on Authentication and Secure Key Management in Internet of Things: Challenges, Countermeasures, and Future Directions. Ad Hoc Networks, 103159.

[18] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, and P. de Geus, ‘‘Malicious software classification using VGG16 deep neural network’s bottleneck features,’’ in Information Technology-New Generations. Cham, Switzerland: Springer, 2018, pp. 51–59.

[19] J. Saxe and K. Berlin, ‘‘Deep neural network based malware detection using two dimensional binary program features,’’ in Proc. 10th Int. Conf. Malicious Unwanted Softw. (Malware), Oct. 2015, pp. 11–20.

[20] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, ‘‘Malware detection with deep neural network using process behavior,’’ in Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC), vol. 2, Jun. 2016, pp. 577–582.

[21] W. Huang, J. W. Stokes, ‘‘Mtnet: A multi-task neural network for dynamic malware classification,’’ in Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment, Cham, Switzerland: Springer, Jul. 2016, pp. 399–418.

[22] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, ‘‘Malware classification with recurrent networks,’’ in 11Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Apr. 2015, pp. 1916–1920.

[23] Sudheer, D., & Krishnan, R. (2019). Multiscale texture analysis and color coherence vector based feature descriptor for multispectral image retrieval. Advances in Science, Technology and Engineering Systems Journal, 4(6), 270-279.

[24] Sivaranjani, R., & Rao, P. M. (2022). Smart energy optimization using new genetic algorithms in Smart Grids with the integration of renewable energy sources. In Sustainable Networks in Smart Grid (pp. 121-147). Academic Press.

[25] Gurumoorthy, S., Rao, B. N. K., Gao, X. Z., Dasari, S. K., Chintada, K. R., & Patruni, M. (2018). Flue-cured tobacco leaves classification: A generalized approach using deep convolutional neural networks. Cognitive science and artificial intelligence: Advances and applications, 13-21.

[26] Devulapalli, S., Potti, A., Krishnan, R., & Khan, M. S. (2021). Experimental evaluation of unsupervised image retrieval application using hybrid feature extraction by integrating deep learning and handcrafted techniques. Materials Today: Proceedings.

[27] H. S. Anderson and P. Roth. (2018). ‘‘EMBER: An open dataset for training static PE malware machine learning models.’’ https://arxiv.org/ abs/1804.04637