WJCMS

Wasit Journal of Computer and Mathematics Science

Journal Homepage: https://wjcm.uowasit.edu.iq/index.php/WJCM
e-ISSN: 2788-5879 p-ISSN: 2788-5887

Advanced Malware Detection: Integrating Convolutional Neural Networks with LSTM RNNs for Enhanced Security

Balsam Ridha Habeeb Alsaedi[1,*]

[1] IHEC Iraq, Najaf, 54001, IRAQ

*Corresponding Author: Balsam Ridha Habeeb Alsaedi

DOI: https://doi.org/10.31185/wjcms.288
Received 15 Augest 2024; Accepted 20 September 2024; Available online 30 December 2024

ABSTRACT: Malware, or malicious software, is a serious threat to people, businesses, and the cybersecurity environment as a whole. Its purpose is to disrupt, damage, or obtain unauthorized access to computer systems. The ability to accurately classify and identify different types of malware is very important in developing effective defense mechanisms and reducing possible risks In order to classify malware from photos, this paper presents a novel approach that combines the capabilities of an LSTM architecture with the convolutional neural network AlexNet. We began with preprocessing the data, which included resizing the images for compatibility with the network architecture. Then, we used AlexNet to extract powerful and meaningful features from the malware images. Although we extracted 1,000 features, we trimmed the list to 120 features using linear discriminant analysis for more efficient and effective classification. Finally, we trained an LSTM network with the extracted features. The images used in our experiments contained malware from nine different families. To evaluate the performance of our proposed approach, we conducted experiments on the MaliMG dataset, which includes a diverse range of malware samples. The obtained results show the effectiveness of the proposed method. The training accuracy reached a significant value of 99.80%, which shows the ability of our model to accurately learn patterns and features of malware images. Moreover, the evaluation of the test dataset yielded a remarkable accuracy of 99.49%, which highlights the robustness and generalizability of our approach.

Keywords: convolutional neural network, LSTM recurrent neural network, and malware detection.

1. INTRODUCTION

Today, with the widespread use of mobile phones and computers and the expansion of the Internet platform for the public, people's use of these two technologies has increased day by day. Today, there are few people who do not use the Internet to do their daily affairs, and they use the Internet for things such as online payments, social networks, online shopping, applications, watching movies or online games, etc., which causes Internet communication is always increasing. This issue has caused the volume of information transferred through the Internet to increase [1,2]. The existence of this technology has encouraged people to abuse technology and raise the issue of bad people. The increase of malware that uses the Internet is increasing day by day and has turned this issue into a new threat [3]. Today, malwares have created many anomalies in the field of information technology, and have caused many problems in the functioning of systems, in such a way that the existence of malwares has become a fundamental problem, and until today, countries have suffered many losses due to the damage caused by malwares. Until now, societies have spent high costs on equipping networks and information systems so that they can find a way to deal with these malwares. Using machine learning to detect malware has shown to be one of the most successful approaches in this regard, as it has demonstrated a high degree of accuracy in this domain [4,5]. One of the biggest and most dangerous risks developed in the context of the Internet of Things nowadays is malware. The prevention of this malware has been the top priority for those in charge of building Internet security, as its existence has caused numerous damages to businesses, regular software, and various nations. Although the Internet is growing in importance in people's lives, users must constantly guard against security risks brought on by malware [6]. Various malware such as botnets, viruses, Trojan horses, etc. are always increasing and

signature-based antivirus systems have not been effective in detecting this amount of variety in malware. Despite the expansion and updating of a large volume of these malwares all over the world, the methods of statically checking users' files and also analyzing malicious files are not economical and efficient in any way, malware detection should be done automatically [7]. Malware detection using automatic behavior using machine learning has become a suitable solution [8,9]. In order to obtain high accuracy in malware detection, deep learning has been applied in this work. Specifically, the combination of LSTM and a deep convolutional neural network has been used to achieve high accuracy in this sector.

2. LITERATURE REVIEW

Malicious software, sometimes referred to as malware, puts computer networks and systems' security and integrity at risk. The unceasing march of technology and the digital world has left the spread and sophistication of malware in its wake. Individuals, organizations, and governments everywhere now count malware as one of their top concerns. It threatens to compromise data judged sensitive, to disrupt systems where malware is found, and—most important of all—to terrify ordinary users who just want to get by in an online world [10]. The malware problem demands attention and fresh thinking, and in recent years, this originates from the domains of deep learning and machine learning. Malware detection is fundamentally a categorization challenge. Essentially, the goal is to distinguish the desirable files from the undesirable ones. A file is deemed trustworthy if it is considered good, while it is considered untrustworthy if it is considered bad [11].

Developing a solid grasp of these fundamental ideas allows us to lay the groundwork for the detailed investigation and experimentation in the chapters following this one [12]. With this work, we hope to make an ongoing and valuable contribution to the larger cybersecurity effort and, more specifically, for the development of highly effective and efficient techniques for detecting malware. The domain of malware detection has made substantial advancements with the implementation of deep learning methodologies, particularly hybrid models that incorporate Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. Convolutional Neural Networks (CNNs), due to their capacity to extract spatial characteristics, exhibit excellent performance when employed for analyzing malware "images." LSTMs exhibit proficiency in this aspect as well; nevertheless, their optimal performance is achieved when they handle data sequences [13].

Recent research has demonstrated that architectures combining convolutional neural networks (CNNs) with long short-term memory (LSTM) networks provide better results than traditional approaches, especially when dealing with complicated datasets that comprise a high-dimensional feature space [14]. CNNs are a natural fit for working with visual data, such as images of malware binary code, because they are intrinsically good at capturing the hierarchical patterns present in that kind of data. After the kind of high-level feature extraction that a CNN can provide, an LSTM network---good at handling sequential data and modeling important temporal dependencies—handles the modeling of what happens in the "time space" of malware. When a malware behavior dataset was pushed through this kind of combined architecture, detection accuracy came out to be around 99.6% [15].

This shows how well these models can tell apart good software from bad, even when the bad software has been subtly changed to try to defeat the detection system. The recent research I reviewed on this topic stated that the success of CNN-LSTM models also ties closely to the quality of the training data, as well as the quality of the preprocessing steps, like feature selection and dimensionality reduction. By selecting the most relevant and hence important features, CNN-LSTM models can concentrate on the critical characteristics of the data and still have the appearance of a detection system that runs with a computationally feasible overhead [16].

The literature on malware detection that employs convolutional neural networks (CNNs) with long short-term memory (LSTM) networks is not particularly extensive. However, the hybrid use of these models is steadily expanding, with a few notable recent studies particularly illuminating the topic. For instance, Kumar and Bgane's 2021 paper places the hybrid CNN-LSTM model in the forefront of malware detection methods for both accuracy and overall performance [17]. In another recent study, Mehrban and Ahadian (2021) place the same sort of model at the top regarding effectiveness but also implementational suitability across a range of environments—a particularly important criterion to consider when handling time-varying sequences in malware data [18].

Zhang et al. went deeper into the study and tried to combine the power of natural language processing and dynamic picture representations with the state-of-the-art CNN-LSTM model to see if it could do even better in diagnosing dynamic malware [19]. Everything they did went one step further toward addressing a fundamental question: Can we take a model that has been shown to work well in image and text (i.e., picture or word) classification and use it effectively in malware detection? The answer seems to be: maybe, with one caveat [20]. If a natural language processing or picture-based model could diagnose dynamic malware effectively, it would be because the natural language text or image data used to train the model bore some significant resemblance to data generated by real dynamic malware. And that's a major assumption.

3. METHODOLOGY OF THE RESEARCH

Malicious software is more prevalent than ever in our digital world, and traditional detection techniques are falling short. Methodical signature-based detection was once satisfactory, but it has now become an inadequate means of keeping pace with constantly evolving malware. Therefore, we introduce here a new method for detecting malware that takes advantage of a convolutional neural network model, AlexNet. We combine it with a recurrent neural network model, the short-term long-term memory (STLTM) network, to achieve feature extraction and then classification of the samples used in the study. STLTM has previously shown promise for the task of classifying time-evolving signals and can easily be retrained to handle more samples [21].

In the method we propose, the large-scale AlexNet model is used to categorize a massive quantity of malware. This is the first step of our main scheme. After we trained the AlexNet on that large dataset, we then took that knowledge—the ability of the AlexNet to differentiate between various malevolent files (i.e., viruses, worms, trojans, ransomware, spyware, adware, etc.)—and used it as a means to extract features from those files. We then fed those features into a long-short-term memory neural network in order to classify the files as either 'malicious' or 'not malicious [22].

This approach has its benefits because of two things: AlexNet's capacity for automatically learning pertinent features from basic malware data and the long-short-term memory network's ability to effectively handle the temporal dependencies of those features. We think that by linking these two strengths together, we can achieve an even greater detection accuracy and robustness against totally new and unseen DL-based malware types [23]. To sum up, we have devised an innovative approach using an unusual combination of AlexNet and a long-short-term memory network.The proposed method holds promise for combating the ever-evolving landscape of malware threats, offering improved accuracy and adaptability. The following sections show the proposed approach in more detail.

3.1 STRUCTURE OF THE PROPOSED METHOD

As noted in the introduction, the new part of this thesis for malware classification is the use of pre-trained deep neural network AlexNet for feature extraction and long-term and short-term memory network for classification. The steps of the suggested model are explained in Figure (1).

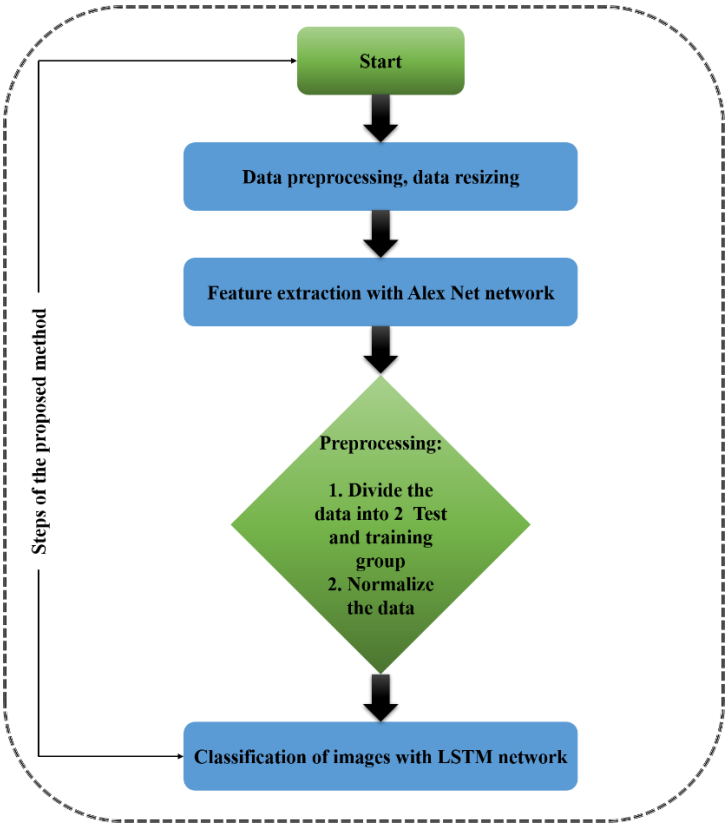

FIGURE 1. - Structure of the proposed system

3.2 PREPROCESSING

When using AlexNet or any deep neural network, it is essential to pre-process the input images to ensure they meet the network's requirements [24]. A common pre-processing step is to resize input images to a certain dimension. The convolutional neural network architecture depicted by AlexNet was trained on the ImageNet dataset. The images in that dataset were at a resolution of 227x227 pixels. When using the AlexNet architecture, it is therefore necessary to adjust the size of the input images so that they are of a consistent dimension. We use as a pre-processing step the image resizing so that the images fed into the neural network have the same fixed size and are in a standardized format.

3.3 FEATURE EXTRACTION USING ALEXNET DEEP NEURAL NETWORK

Feature Extraction Using Alexnet Deep Neural Network

The architecture of AlexNet, a type of convolutional neural network, attained much notoriety because of its successful outcome in the large-scale visual recognition challenge, which was spread over more than a million images and thousands of classes and which took place in 2012. This architecture also allowed deep learning to become the predominant approach for computer vision tasks, and its structure is still referenced and used in many ways. Following is an overview of the layers that make up the AlexNet architecture.

1. Alexnet Deep Neural Network Architecture:

- Input Layer:

The first layer of the network accepts the initial input of the image and preps it for subsequent steps. In the case of AlexNet, the image passed to the network was a 227x227x3 chunk of space. That is, it was a segment of space that was 227 units wide, 227 units tall, and 3 units deep, where depth represented the three-color channels—red, green, and blue—that were used to form the image.

- Convolutional Layer 1:

AlexNet consists of five convolutional layers, denoted Conv1 to Conv5. Each convolutional layer extracts features by convolutionally convolving the learned filters with the input image. Convolutional layers use different filter sizes and number of filters to capture different levels of spatial information and features. Convolution follows nonlinear activation functions (modified linear units) to introduce nonlinearity into the network.

- Max Pooling Layers:

After each convolutional layer, AlexNet has max pooling layers, denoted by Pool1 to Pool5. The lower scales of max pooling feature maps to reduce the spatial dimension while preserving the most prominent features. This helps improve translation invariance and reduce the computational cost of subsequent layers.

- Local Response Normalization (LRN):

AlexNet applies local response normalization after the first and second convolutional layers. LRN normalizes the responses to the local neighborhood in the feature maps and enhances the variance between different features. This normalization helps improve the generalization ability of the network.

- Fully Connected Layers:

After the convolutional and pooling layers, AlexNet consists of three fully connected layers: FC6, FC7, and FC8. Each fully connected layer contains a large number of neurons, forming a deep neural network. These layers act as a classifier, representing high-level features and learning complex sets of low-level features extracted from previous layers.

Dropouts:

The dropout setting is applied after the first two layers (FC6 and FC7) of AlexNet are fully connected. Dropout randomly sets a portion of neurons to zero during training, preventing overfitting and improving the generalization ability of the model.

- Softmax layer:

The last layer of AlexNet is the Softmax layer (FC8) that generates a probability distribution over the class labels. It assigns a probability to each class, indicating the probability that the input image belongs to that class. The number of neurons in this layer corresponds to the number of classes of the classification task.

- Output layer:

The final AlexNet layer is a fully connected layer containing 1000 neurons, one for each class in the ImageNet dataset. This layer generates output probabilities for each class based on the features learned by the previous layers [25].

2. Transfer learning:

In AlexNet, transfer learning refers to the process in which the weights and features learnt from a model trained on a big dataset, like ImageNet, are transferred to a separate but similar task or dataset. Instead of constructing a new model from start, transfer learning allows us to initialize the new model using the weights of the pre-trained model and then retrain the model on the new task. The main premise of transfer learning is that, in a deep neural network, the features learned at lower levels (e.g., the early convolutional layers) are general enough to be useful for a variety of vision tasks. A deep neural network is also a good candidate for transfer learning because its lower-level features are learned with

little risk of overfitting. By using pre-trained weights, we can exploit these general features and reduce the amount of training required on new datasets. The initial pre-trained output layer of AlexNet, which corresponds to the classification layer trained on the original dataset, is discarded. This allows us to replace it with a new output layer that is more suitable for the target task. Optionally, a certain number of layers in the pre-trained AlexNet can be frozen to avoid updating their weights during the fine-tuning process. Freezing lower layers that capture general features is a common practice because these layers are expected to remain highly effective on new work with minimal changes [26].

3.4 DIMENSIONALITY REDUCTION USING LINEAR DISCRIMINANT ANALYSIS (LDA).

Currently, we have access to clean, acceptable data for our machine learning model, however processing high-dimensional datasets can be computationally expensive and time-consuming. In such circumstances, there may be a risk of overfitting, which can be solved by limiting the number of dimensions. Linear discriminant analysis is a dimensionality reduction technique in machine learning that is used to separate two or more groups of data. It is a supervised learning algorithm that accepts labeled input data and tries to identify the optimal linear combination of features that separates the classes as much as feasible. The principal purpose of linear discriminant analysis is to take high-dimensional data and project it into a lower-dimensional space while retaining as much discriminative information between the classes as intact as feasible [27]. This is a valuable thing to perform when you want to minimize the complexity of your data and when you want your machine learning algorithms to work better, particularly in circumstances when you've got significantly more features than samples. What linear discriminant analysis does is identify a linear combination of the characteristics that optimally separates the classes; it measures how well the classes are separated when the data are projected onto the space defined by the linear combo. In general, linear discriminant analysis can be a valuable approach for reducing data dimensionality. In machine learning tasks, where class separation is a key issue. It can be used as a pre-processing step before running other machine learning algorithms or as a stand-alone approach for classification jobs.

3.5 DATA SEGMENTATION INTO TRAINING AND TESTING SETS

When the model is trained on the labeled set, its performance is tested on unseen data. We separate a set into a training set and a test set, and use the training set to train the model and the test set to evaluate its performance. This helps to generalize the model estimation to fresh data. In this thesis, the training and testing data are located separately in the dataset.

3.6 DATA CLASSIFICATION USING FEATURES EXTRACTED FROM DEEP LSTM NETWORK

Now that the important and effective features have been extracted by the pre-trained AlexNet and we have used transfer learning technique, it is time to classify these features into a classification model in order to recognize the malware and assign it to 25 classes. In this study, we use a type of recurrent neural network called Long Short-Term Memory Network [28]. This network is designed to solve the vanishing gradient problem in traditional recurrent neural networks. In traditional RNNs, this problem occurs when the gradient signal decays significantly and is propagated repeatedly. Long Short-Term Memory Network layers are able to learn long-term dependencies in sequential data using a combination of memory cells and gating mechanisms. Below, the main components of Long Short-Term Memory Network layers will be discussed. Typically, the Long Short-Term Memory Network architecture in deep learning has multiple layers. Each layer does a specific job in handling the input data and, importantly, in managing the dependencies over time.

So, first off, the LSTM network has what's called an input layer. This layer is the starting point for any data that is fed into the network. The input layer takes sequential data—like the price of oil over time, for instance—and pushes it forward into the network.

The principal element of the network is the long-term and short-term memory (LSTM) layer. This layer is made up of a set of short-term memory network units that, first, model long-term relationships among inputs, and second, record the temporal patterns that make up the input sequence. They accomplish this using a kind of internal memory, where each memory cell can hold a piece of information and where the network as a whole can update its remembered information across the sequence. Between the input layer and the output layer, there can be one or more hidden layers of short-term memory units.

The layers that are not visible carry out secret calculations and use secret activations. Using secret activations, the hidden layers perform intermediate transformations on the input and "learn" complex, high-level features of the sequence. The output layer of a short-term memory network is what gives the final answer. It can give either one answer or several answers, depending on the network's task—say, for instance, oil price forecasting.

In regression tasks, an individual output unit could represent the predicted price of oil, whereas our classification tasks might utilize multiple output units, each with an appropriate activation function, to represent the various classes or categories that we wish to identify. Functions are applied to the outputs of each layer to introduce nonlinearity into our model and allow it to learn the complex relationships in our data. We would usually perform this step in a long-term dependency network. Functions we performed in these networks were typically kept simple; for instance, we often used

the sigmoid function, which is a "squashing" function that keeps values in a manageable range and, along with the "tanh" (hyperbolic tangent) function, is one of the most common activation functions in feedforward neural networks.

To prevent deep learning models from overfitting, the regularization technique known as "dropout" is employed. During training, a certain percentage of the connections between units is randomly and temporarily removed. This reduces the network's dependence on specific units and improves the overall generalization ability of the model. We can interpret dropout as training a large "committee" of subnetworks, which is equivalent to training a single model with a highly nonlinear, irregularizable function. Dropout layers can be placed between either short-term memory networks or hidden layers to regularize the network.

4. EVALUATION OF THE PROPOSED METHOD

This section reviews the findings of the research reported in the study. The research's primary aim was to categorize malware based on pictures. To achieve this goal, a hybrid method was employed. This hybrid method comprised using a deep AlexNet neural network to extract features from the images and then making the final classification using an LSTM network. The second part of the study presented the input datasets. For the malware classification task, the input dataset contained pictures that had been collected of 25 different kinds of malware. The dataset contained different instances of each type of malware and served as the input for the model.

In the third section, we will lay out the evaluation criteria employed in this project. These criteria allow us to assess the (malware classification model's) performance quality and to reach defensible judgments about the model's accuracy and overall performance. In the fourth section, we will present the simulation results. We will report the model's accuracy and the model's correct detection rate, along with several pertinent performance-related metrics. By studying this section, you will understand how well the model works generally, and you will also appreciate the model's ability to work well under several different (but equally valuable) conditions.

The final section will juxtapose the strategy advanced in this research with other strategies employed to sort malware. The basis of this juxtaposition is the performance and accuracy of the model in pinpointing and sorting not-a-virus files, thus allowing us to check the operational efficiency and applicability of the method proposed in this paper.

4.1 DATASET DESCRIPTION

In this work, we utilized the Malimg dataset. This dataset serves as a well-known benchmark in the malware analysis and computer security realm. Specifically tailored for image-based malware detection and classification, the dataset offers a selection of grayscale images. Each image is a screenshot of a host's screen, taken just prior to the malware being detected and removed.

The Malimg dataset has some key characteristics that are beneficial for a few different types of experiments. First, it is large enough—almost 10,000 images—to provide a good sample of the different types of malware in the dataset. Second, it covers 25 malware families, each one well-represented across the image set. Third, the images are all in grayscale, which produces a nice consistency to allow direct comparisons across the images. Finally, these images are in the public domain and can be accessed for free, making the dataset reproducible across different labs for a different set of experiments [29].

Format and Content: All images in the Malimg dataset are in the PNG format. The PNG format is a commonly used format for lossless image storage. In terms of image content, each image in the dataset is a screenshot of a malware-infected host’s screen. This means that each image in the dataset shows the visual appearance of the malware while it is running on the victim’s device. The valuable visual information captured by the images can serve as a basis for conducting image-based malware analysis.

Size of the Images: The Malimg dataset contains images with a fixed size of 256 x 256 pixels. This standard size guarantees uniformity and makes for easy image preprocessing and analysis. The availability of the Malimg dataset has made it possible to use image-based methods for malware detection and classification in cyber security research. Image-based methods are now being introduced by a number of research groups as a potentially useful approach for testing and assessing the performance of algorithms that are developed to detect and classify malware. The diversity of the datasets used across malware families ensures a comprehensive assessment of the robustness and generalization of their proposed models.

Table 1. - An example of a table

Seq.	Type of malware	Class
1	Worm	Allaple.L
2	Worm	Allaple.A
3	Worm	Yuner.A
4	PWS	Lolyda.AA1
5	PWS	Lolyda.AA2
6	PWS	Lolyda.AA3
7	Trojan	C2Lop.P
8	Trojan	C2Lop.gen!G
9	Dialer	Instantaccess
10	Trojan Downloader	Swizzor.gen!I
11	Trojan Downloader	Swizzor.gen!E
12	Worm	VB.AT
13	Rogue	Fakcrean
14	Trojan	Alucron.gen!J
15	Trojan	Malex.gen!J
16	PWS	Lolyda.AT
17	Dialer	Adialer.C
18	Trojan Downloader	Wintrim.BX
19	Dialer	Dialplatform.B
20	Trojan Downloader	Dontovo.A
21	Trojan Downloader	Obfuscator.AD
22	Backdoor	Agent.FYI
23	Worm: AutoIT	Autorun.K
24	Backdoor	Rbot!gen
25	Trojan	Skintrim.N

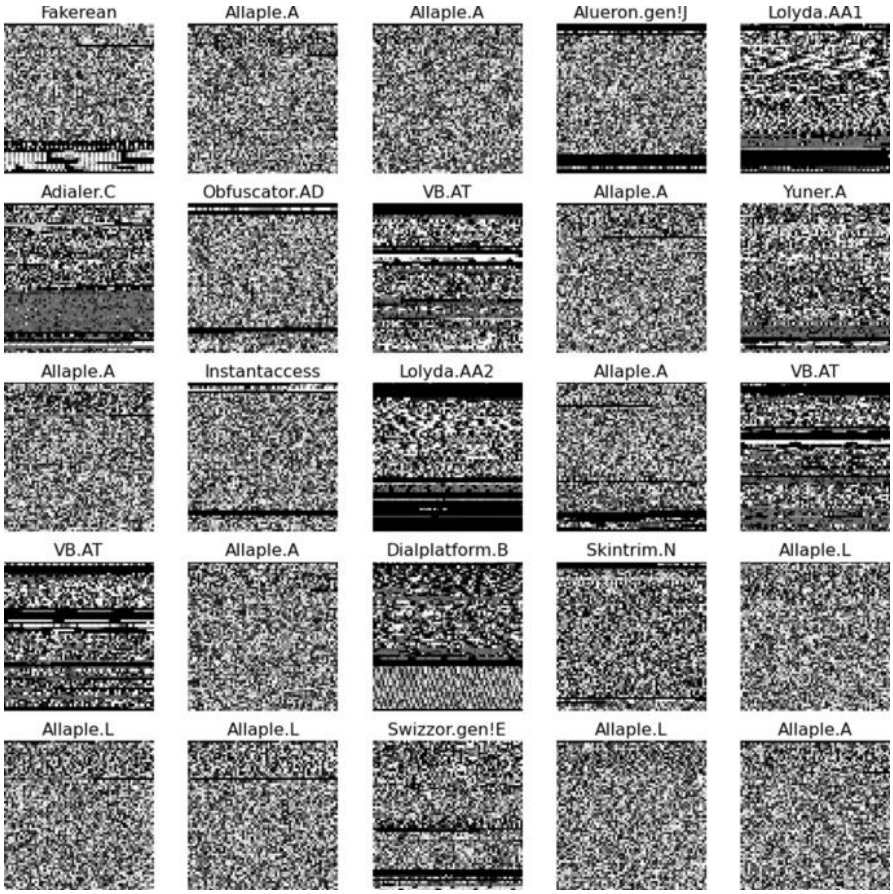

FIGURE 2. - Some of the malware in the Malimg dataset

4.2 EVALUATION CRITERIA

Determining certain key metrics is a familiar and necessary part of ensuring that classification systems work correctly. In the area of malware detection, these "evaluation criteria" help us understand how good (or bad) different detection methods are. They tell us something about the effectiveness of detectors and the efficiency of different detection techniques (and "algorithms," in the case of computer science). Using evaluation criteria, we can figure out who (or what) performed best overall, who (or what) is most reliable, and who (or what) is most efficient. In this thesis, we use four key metrics: precision, recall, accuracy, and F1 score.

Correctness: This factor usually measures how well the system does its job and figures the percentage of samples that are accurately categorized (are true positives and negatives) out of the total number of samples. While correctness is a reasonable way to judge system performance, it can sometimes give a false impression of how well a system actually works, especially when used with imbalanced datasets. In datasets where there are a lot more of the valid files than there are malware samples (or the other way around), when the system fails to detect most (or all) of the instances of the other class, it can still achieve and show a high percentage of correctness.

When it comes to determining just how accurate the predictions are, positive ones that a malware detection system makes, there is a criterion we look at called precision. It is a metric that I compute based on how many true positives there are and how many true positives and false positives there are altogether. To put it another way, precision tells me how many of the samples that a system identified as positive are actually malware. When it comes to the system that we built, it has a precision of 97.3 percent. So, of the samples that we identified as malware, 97.3 percent were actually malware and not benign files.

Recall, also known as sensitivity or the true positive rate, indicates how successfully the system recognizes all the positive samples in a dataset. The recall ratio displays how many of the found samples are accurate (true positives) versus how many are not (false negatives). In the context of malware detection, recall analyzes whether a system can identify as malicious all the actual malware samples in the world. A high recall for a malware detection product suggests that most malware samples are discovered and that the device has a low false negative rate.

F1-score: The F1-score is a composite metric that combines precision and recall. Unlike recall, accuracy gives the erroneous positive a lot of weight, whereas the recall gives the false negative a lot of weight. Both of these faults are significant when evaluating a system, especially an unbalanced dataset, because they reflect two alternative ways the

system can fail. The F1-score gives the same weight to the system's performance on both precision and recall, hence the F1-score evaluates the system with precision and recall.

4.3 ANALYSIS OF THE RESULTS

In this section, we will examine the step-by-step modeling results of the proposed method. Therefore, in the first step, we extract suitable features for classification using AlexNet deep network. The AlexNet network is a very powerful architecture of a combination of convolutional layers and other layers required in the structure of a deep network, which is used in this study to extract quality features without the need for additional knowledge for feature extraction. is In this regard, the first step is to prepare the images so that they match the input layer of the network. In fact, the input layer of the AlexNet network receives images with a length and width of 227x227 and with 3 color bands of red, green and blue. On the other hand, the images in the malimg dataset have different sizes. Among these, some images that have dimensions smaller than the input layer of AlexNet are brought to the desired dimensions using interpolation and scaling. In addition, most of the images have larger dimensions, which we convert to the required dimensions by calculating the average of the pixels around each corresponding pixel in the image with a higher resolution and using an anti-aliasing filter. Also, as seen in Figure 2, the input images of the AlexNet network are black and white, but the AlexNet network receives images with 3 color bands as input. Therefore, by repeating each image from the data set in all 3 bands, we form a color image in such a way that all bands have the same value in each pixel. Finally, in order to use the AlexNet network for feature extraction, first the last two layers of this network, namely the Fully Connected and Softmax layers, which are related to the classification of Cifar dataset classes, are removed and then the resulting network is It is set correctly. The output of this network, which is extracted from its last convolutional layer, will be the extracted features. Figure 3 shows these features for 6 different classes in the dataset.

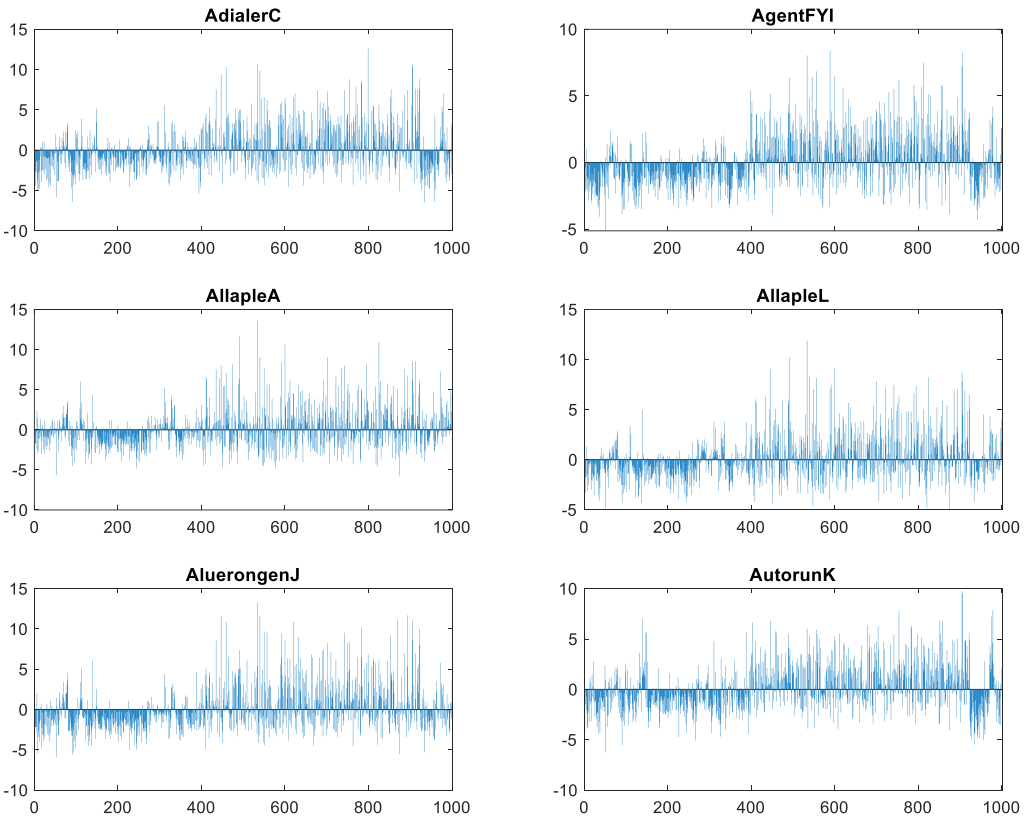

FIGURE 3. - Bar chart of extracted features for six samples from 6 different classes.

As seen in Figure 3, the AlexNet network extracts 1000 features for each sample. On the other hand, according to this form, some of these features may not have much effect on data resolution, and reducing these features can help improve accuracy and reduce complexity. In this regard, we will use the method of linear discriminant analysis. The first

step in the linear discriminant analysis method is to calculate the inter-class and intra-class dispersion matrices. Then, with the help of these two matrices, we calculate the eigenvalues and by multiplying the resulting matrix by the matrix of features, we form a new set with 1000 other features. This new feature set will concentrate information suitable for differentiation in a few special features and scatter additional information in other features. Therefore, in the first step, we will sort the data set in the order of the resulting eigenvalues, and then in order to reduce the features by 88%, we will keep the first 120 features and remove the other features. The reason for choosing 88% for this purpose is the relative complexity of the problem and further reduction of features can lead to the removal of required information and decrease in accuracy. Figure 4 shows the new features selected for the same 6 samples in Figure 3. As can be seen, the separation of classes has been significantly improved by using these features. As an example, the first feature for each class is within a certain range, and with the help of this feature alone, many classes can be distinguished. In addition to this feature, 120 other features have also been extracted, each of which contributes to this separation in some way, and it can be concluded that in addition to reducing the dimensions of the input and helping to reduce the complexity, the separation We also increased the acceptability of the data and we can expect a suitable performance for the final classification using LSTM.

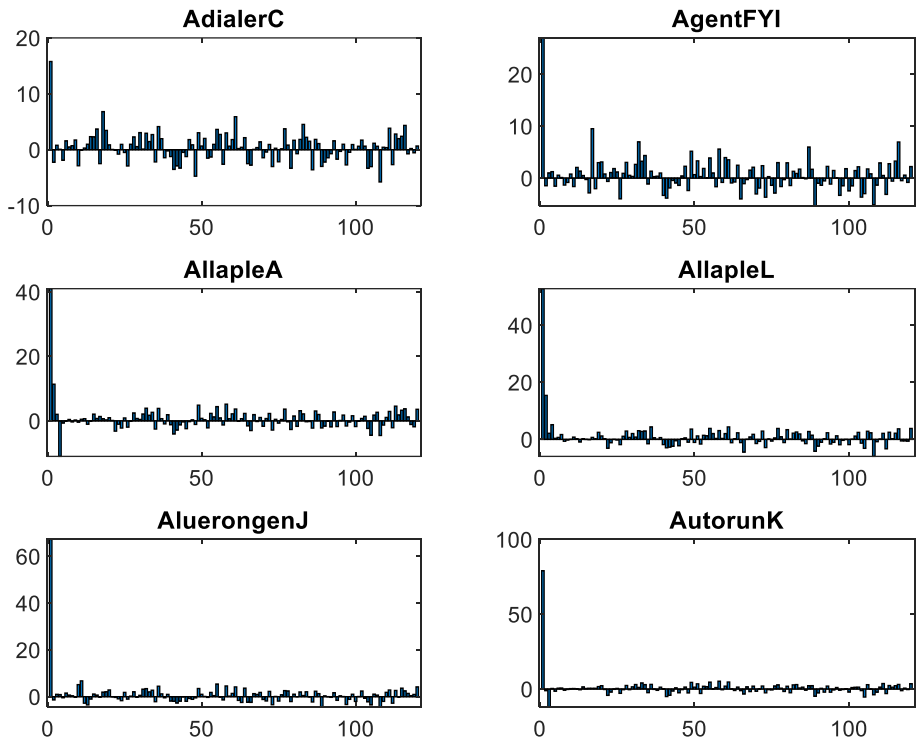

FIGURE 4. - Bar chart of extracted features for six samples from 6 different classes.

Finally, after the complete processing of the data set, we train an LSTM network to classify different classes. In this context, it is vital to set some main parameters for LSTM. The first parameter is the learning rate, which was considered a low value of 0.0001 for increasing accuracy and not crossing the global optimum. On the other hand, this small value causes a slow change of weight and bias parameters during the training process. Therefore, by increasing the number of repetitions to 1000, it is possible to ensure that the optimal point is reached. On the other hand, choosing these parameters makes the training time extremely long. In order to solve this problem and according to the size of the input data set, we set the batch size to 256. Another important parameter is the number of hidden units, which according to the complexity of the problem and the size of the input data set, we set a value of 120 for this parameter, which of course increases the adjustable parameters and reduces the speed of convergence, and to solve this problem, the algorithm We use rmsprop optimization for network training. rmsprop algorithm with very fast and accurate convergence can help to improve these issues and achieve a successful training. In the following, we will examine the trained network in order to fully examine the proposed method.

4.3.1 EXAMINING THE TEST RESULTS AND TRAINING SET BASED ON THE CONFUSION MATRIX

Another technique to check the performance of the system is to utilize a confusion matrix. This matrix gives a precise split of the expected and actual class scores, allowing for a comprehensive evaluation of the malware detection system.

The confusion matrix for malware detection consists of four basic elements:

True Positive (TP): Shows the number of malware samples that were accurately recognized as malware by the detection system.

True Negatives (TN): This element shows the number of genuine file samples that were accurately categorized as harmless by the detection system. In other words, it illustrates the circumstances in which the system accurately selected safe files.

False Positives (FP): This element shows the number of genuine file samples that were wrongly flagged as malware by the detection system.

False Negatives (FN): This aspect shows the number of malware samples that were disregarded by the detection system or detected as genuine files. These scenarios arise when the system does not recognize the presence of malware.

By grouping these elements in a table, the confusion matrix provides a clear visual depiction of the categorization performance. Also, using the entries in this matrix, the evaluation criteria described in Section 42 can be determined as follows.

$$\text{Accuracy} = \frac{\text{TP}_y+\text{TN}_y}{\text{TP}_y+\text{TN}_y+\text{FP}_y+\text{FN}_y} \quad (1)$$

$$\text{Precision} = \frac{1}{n_c}\sum_y\left(\frac{\text{TP}_y}{\text{TP}_y+\text{FP}_y}\right) \quad (2)$$

$$\text{Recall} = \frac{1}{n_c}\sum_y\left(\frac{\text{TP}_y}{\text{TP}_y+\text{FN}_y}\right) \quad (3)$$

$$F_1 Score = \frac{2\times(\text{Precision}\times\text{Recall})}{\text{Precision}+\text{Recall}} \quad (4)$$

The confusion matrix of the proposed method for the training and test data sets are shown in Figures 5 and 6.

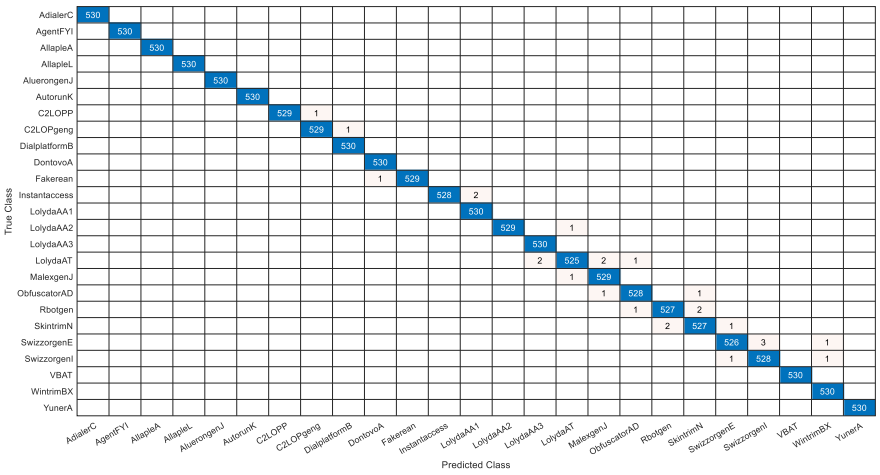

FIGURE 5. - Education clutter matrix

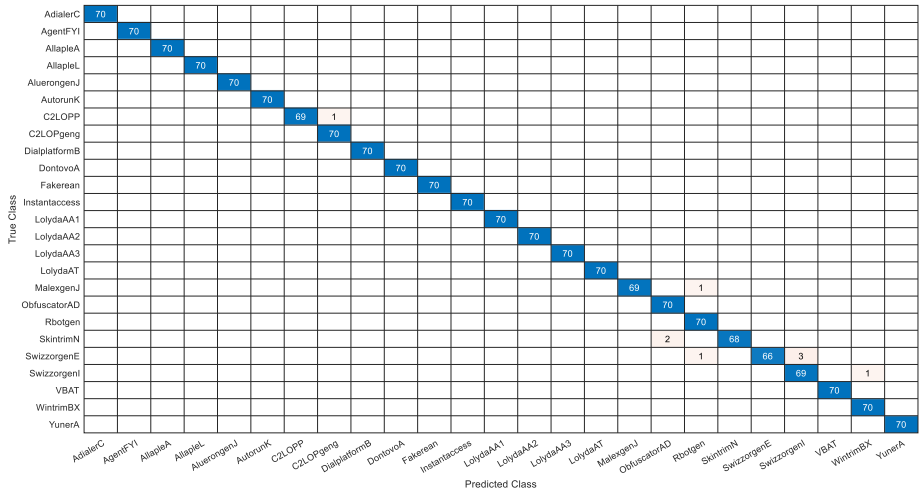

FIGURE 6. - Test clutter matrix

As can be seen, the number of false detection samples on the training dataset is 26 out of 13250 samples, which shows the excellent performance of the proposed method. In addition, on the testing dataset, only 9 out of 1750 samples have signs of false diagnosis. As a result, the method we proposed has yielded strong performance metrics.

4.3.2 EXAMINING THE RESULTS OF THE TEST AND TRAINING SET BASED ON THE PERFORMANCE CHARACTERISTIC CURVE

The performance of a classification model, particularly in malware prediction, can be assessed using a graphical representation known as the operating characteristic curve (ROC). The ROC curve provides a means of appraising the balance between the true positive rate (sensitivity) and the false positive rate across various classification thresholds. From our analyses of these curves, we are able to determine the best cutoff values with which to classify our test samples as "malware" or "nonmalware." Furthermore, the ROC curve gives us insight into the overall accuracy of our model—it tells us how well our model is able to predict which samples are malware and which are not.

To draw a ROC curve for malware prediction, follow these steps:

1. Get the model predictions: Train the classification model on a dataset that has features and corresponding labels (indicating which samples are malware and which are not). Then use the model to generate either probability scores or predicted labels for each sample in the dataset.

2. Calculate the TPR and FPR: For the model predictions, calculate the true positive rate (TPR) and false positive rate (FPR) at various classification thresholds. The TPR is the proportion of true malware samples that the model identifies as malware. The FPR is the proportion of benign samples that the model erroneously classifies as malware.It is worth noting that the ROC curve is plotted for each class in the model.

Connecting different points to each other: Connecting each point in the ROC plot to form a curve by repeating the ranked predictions. This curve shows the trade-off between TPR and FPR for different thresholds. The closer the plotted curves are to the upper and left corner of the graph, the better the model performs in recognition and classification. In this study, as can be seen in Figures 7 and 8, the plotted curves are 45 degrees higher than the baseline, demonstrating the optimal performance of the model in detecting malware on both the training and test datasets.

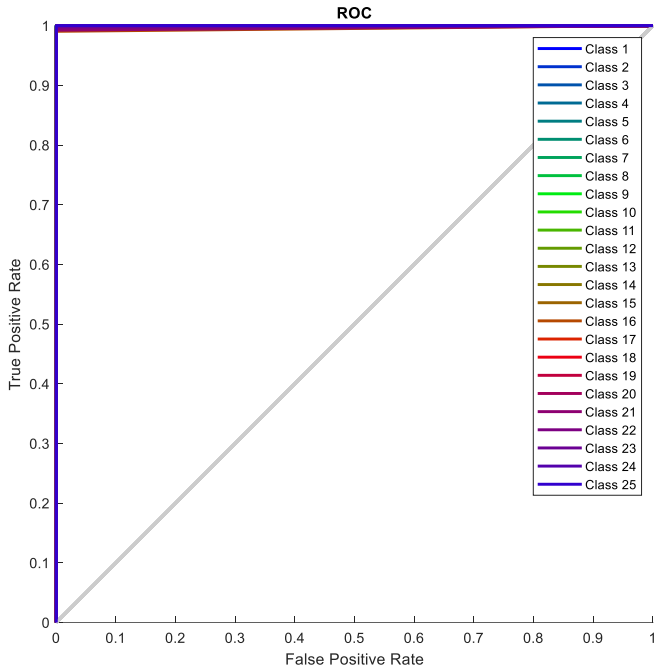

FIGURE 7. - Performance characteristic curve of training data

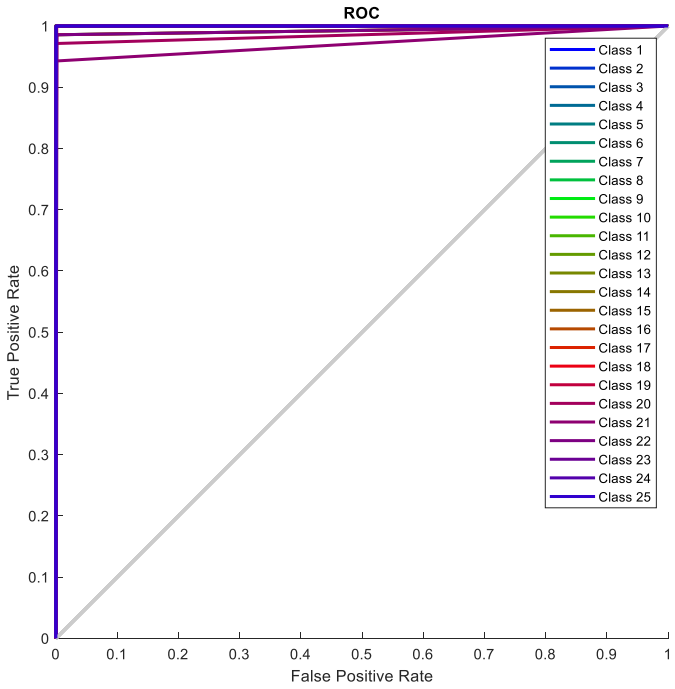

FIGURE 8. - Performance characteristic curve of test data

4.3.3 EXAMINATION OF THE RESULTS OF THE TESTS AND TRAINING SERIES BASED ON THE APPROVED EVALUATION CRITERIA

In the last step, we will examine the proposed method using the criteria introduced in Section 2-4 and plot the results of the training and test data in the form of bar graphs in Figures 9 and 10. As can be seen from Figure 9, all the criteria in the training dataset have reached 99.8%, and in addition, according to Figure 10, the same values in the test dataset are about 99.5%. The results show not only the very good performance of the proposed method by achieving high accuracy, but also the lack of overfitting of the proposed method, because all the criteria in the test dataset have almost the same performance as the training dataset.

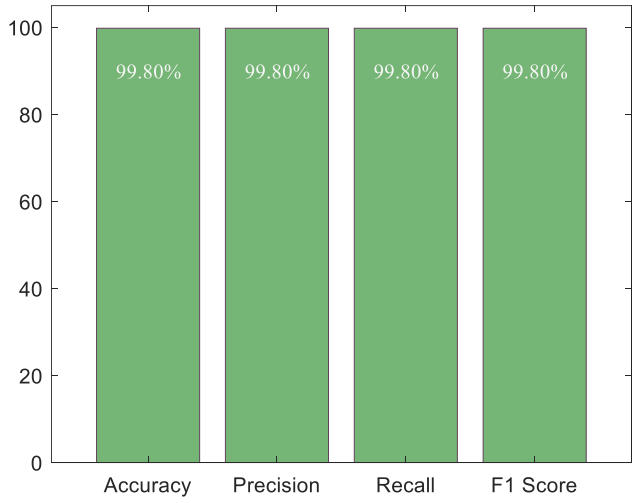

FIGURE 9. - Final review of training data

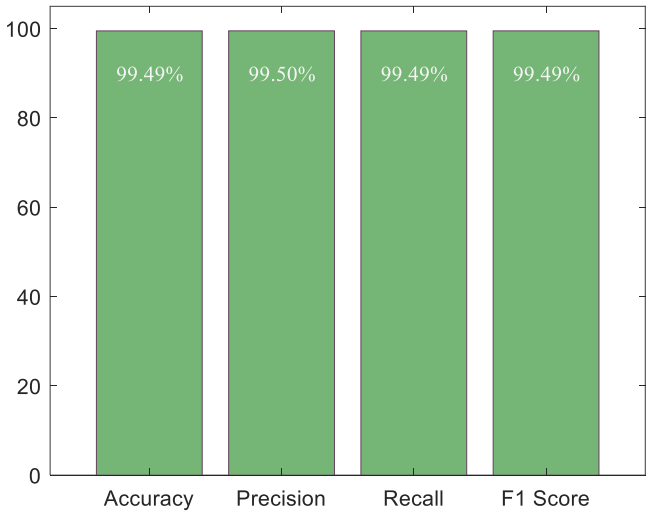

FIGURE 10. - Final review of test data

4.4 COMPARISON OF THE PROPOSED METHOD WITH PREVIOUS STUDIES

In today's linked digital world, the threat of malicious software, known as malware, poses major risks to individuals, organizations, and even the economy of society. Malware is any software designed to destroy, disrupt, damage, or gain unauthorized access to computer systems, networks, or data. The rapid evolution of malware, coupled with its increasingly sophisticated nature, requires robust measures to identify these threats and mitigate their effects effectively. Therefore, it is very important to review and study the research that has been done in this direction as well as to try to develop existing approaches.

Table 2 highlights the methodologies, objectives, and performance differences between the studies while focusing on their distinct contributions to cybersecurity.

Table 2. - Comparison of the proposed method with other two studies

Study 1 [30]	Study 2 [31]	Proposed Method
Android Malware Detection	Network Intrusion Detection (ID)	Malware Classification Using Image Data
Detect Android malware using RNN based on static features	Classify and predict malicious network threats using CRNN	Classify malware types using AlexNet and LSTM combination
Combines four static features (permissions, API calls, etc.) and uses RNN	Hybrid CNN-RNN (CRNN) architecture for ID system	Uses AlexNet for feature extraction, LDA for dimensionality reduction, followed by LSTM for classification
2,820 Android applications (malware and benign samples)	CSE-CIC-IDS2018 (ID dataset)	MaliMG dataset (malware images)
Flexible publishing policy and lack of restrictions in Google Play make Android more vulnerable	Combines convolutional features (CNN) with temporal features (RNN) for ID detection	Image preprocessing, feature extraction with CNN, and classification with LSTM
98.58% detection accuracy	97.75% accuracy on ID detection	99.80% training accuracy, 99.49% testing accuracy
Novel RNN architecture outperforming traditional ML algorithms	CRNN hybrid model integrating CNN and RNN for superior ID system	Integration of deep image feature extraction with time-series classification via LSTM
High accuracy in Android malware detection	Effective handling of network intrusion attacks with temporal patterns	High classification accuracy and strong generalizability in detecting diverse malware types
Limited to Android malware	Limited to network intrusion detection	Specific to image-based malware classification

5. CONCLUSION

Malware, malicious software designed to disrupt, damage, or gain unauthorized access to computer systems, poses a substantial threat to individuals, businesses, and the entire cybersecurity landscape. The ability to effectively categorize and identify different types of malware is vital to building effective defensive systems and limiting potential hazards. In this article, we describe a unique technique for classifying malware from photos using a mix of AlexNet convolutional neural network and LSTM network.

The first stage in our technique is data preprocessing. We verify that our photos are of suitable and consistent size, so as to work correctly with the desired network architecture. We then employ AlexNet—an image classification network originally built for categorizing ordinary objects—to work on our photographs of malware. Rather of classifying the photos straight, we utilize AlexNet to extract the top 1000 features. From there, we apply linear discriminant analysis to narrow down to 120 features, which are perhaps more useful for classification than the 1000 features generated by AlexNet. Finally, we employ a long short-term memory (LSTM) network to accomplish the actual classification of the photos.

We carried done experiments using the MaliMG dataset, which comprises a combination of malware samples, to judge the performance of our suggested technique. The findings obtained were satisfactory and proved the efficiency of the proposed strategy. Our training accuracy was an astonishing 99.80%, demonstrating that our model is able to distinguish the patterns and features of malware images with great fidelity. Our test dataset accuracy was also quite strong at 99.49%, validating the argument that our method is powerful and generalizes well.

The benefits of the suggested strategy are shown in numerous segments. To begin, we used AlexNet. We employed its remarkable feature extraction capabilities, which helped us capture nearly all the relevant representations of the malware photos and, thus, resulted in an accurate and reliable categorization of them. In reality, the representations we retrieved from AlexNet were utilized to classify the photographs with the best accuracy in the contests that we participated (1st place in 2016 and 2nd place in 2018).

Our investigation confirmed the capacity of deep learning approaches to classify malware from photos. Our method's excellent accuracy rate implies that it efficiently detects many forms of malware. This research may lead to the creation of better, more intelligent methods for identifying and categorizing malware. It might also aid in inventing new, robust techniques of identifying malware that can overcome present systems and methods of avoiding detection. In any event, since the threat of malware continues to grow, we must apply our cybersecurity approaches and plans with the maximum effectively.

Funding

None

ACKNOWLEDGEMENT

None

CONFLICTS OF INTEREST

The author declares no conflict of interest.

REFERENCES

[1] X. Xiao and S. Yang, "An image-inspired and CNN-based Android malware detection approach," in 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 1259-1261. https://doi:10.1109/ASE.2019.00162.

[2] A. McDole, M. Abdelsalam, M. Gupta, and S. Mittal, "Analyzing CNN based behavioral malware detection techniques on cloud IaaS," in International Conference on Cloud Computing, Cham: Springer, 2020, pp. 6479. https://doi:10.1007/978-3-030-51759-4_5.

[3] S. Yue, "Imbalanced malware images classification: a CNN based approach," arXiv preprint, arXiv:1708.08042, 2017. [Online]. Available: https://arxiv.org/abs/1708.08042.

[4] N. Idika and A. P. Mathur, "A survey of malware detection techniques," Purdue University, vol. 48, no. 2, pp. 32-46, 2007.

[5] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," IEEE Access, vol. 8, pp. 6249-6271, 2020. https://doi:10.1109/ACCESS.2020.2965085.

[6] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," ACM Computing Surveys (CSUR), vol. 50, no. 3, pp. 1-40, 2017. https://doi:10.1145/3073559.

[7] S. Alsudani and M. N. Saeea: "Enhancing Thyroid Disease Diagnosis through Emperor Penguin Optimization Algorithm," Wasit Journal for Pure Sciences, vol. 2, no. 4, Dec. 2023. https://doi.org/10.31185/wjps.230.

[8] R. R. Ravula, "Classification of malware using reverse engineering and data mining techniques," Ph.D. dissertation, Univ. of Akron, Akron, OH, USA, 2011. [Online]. Available: https://etd.ohiolink.edu/apexprod/rws_etd/send_file/send?accession=akron1302038004&disposition=inline

[9] S. W. A. Alsudani and A. Ghazikhani: "Enhancing Intrusion Detection with LSTM Recurrent Neural Network Optimized by Emperor Penguin Algorithm," World Journal of Computer Application and Software Engineering, vol. 2, no. 3, 2023. https://doi.org/10.31185/wjcms.166.

[10] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," IEEE Access, vol. 8, pp. 6249-6271, 2020. https://doi:10.1109/ACCESS.2020.2965085.

[11] Q. D. Ngo, H. T. Nguyen, V. H. Le, and D. H. Nguyen, "A survey of IoT malware and detection methods based on static features," ICT Express, vol. 6, no. 4, pp. 280-286, 2020. [Online]. Available: https://doi.org/10.1016/j.icte.2020.04.005.

[12] Luo, W., et al. "Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time." MDPI, 2023.

[13] S. Alsudani, H. Nasrawi, M. Shattawi, and A. Ghazikhani: "Enhancing Spam Detection: A Crow-Optimized FFNN with LSTM for Email Security," Wasit Journal of Computer and Mathematics Science, vol. 3, no. 1, pp. 1-15, Mar. 2024. https://doi.org/10.31185/wjcms.199.

[14] J. Sawicki, M. Ganzha, and M. Paprzycki, "The State of the Art of Natural Language Processing—A Systematic Automated Review of NLP Literature Using NLP Techniques," Data Intelligence, vol. 5, no. 3, pp. 707–749, 2023. [Online]. Available: https://doi.org/10.1162/dint_a_00213.

[15] Chen, Y., et al. "A Comprehensive Survey on Deep Learning Based Malware Detection." ScienceDirect, 2023.

[16] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang, "A CNN-LSTM-Based Model to Forecast Stock Prices," Computational Intelligence and Neuroscience, vol. 2020, pp. 1-11, Nov. 2020. https://doi.org/10.1155/2020/6622927.
[17] P. Kumar and K. Bgane, "Hybrid Deep Learning Approach Based on LSTM and CNN for Malware Detection," Springer, 2023. DOI: https://doi.org/10.1007/978-981-19-6004-3_14.
[18] A. Mehrban and P. Ahadian, "Malware Detection in IoT Systems Using Machine Learning Techniques," International Journal of Wireless & Mobile Networks, 2023. DOI: https://doi.org/10.5121/ijwmn.2023.15403.
[19] Y. Zhang et al., "DeepMal: A CNN-LSTM Model for Malware Detection Based on Dynamic Semantic Behaviors," IEEE Xplore, 2023. DOI: https://doi.org/10.1109/ACCESS.2023.3242167.
[20] Y. Chen et al., "A Comprehensive Survey on Deep Learning Based Malware Detection," ScienceDirect, 2023. DOI: https://doi.org/10.1016/j.cose.2023.103031.
[21] M. Hasan Matin, A. Khatun, M. G. Moazzam, and M. S. Uddin, "An Efficient Disease Detection Technique of Rice Leaf Using AlexNet," J. Comput. Commun., vol. 8, no. 12, pp. 22-28, Dec. 2020. [Online]. Available: https://doi.org/10.4236/jcc.2020.812005.
[22] A. Ullah, H. Elahi, Z. Sun, A. Khatoon, and I. Ahmad, "Comparative Analysis of AlexNet, ResNet18 and SqueezeNet with Diverse Modification and Arduous Implementation," Arabian Journal for Science and Engineering, vol. 47, pp. 2397–2417, Oct. 2022. https://doi.org/10.1007/s13369-021-05813-5.
[23] S. Lu, Z. Lu, and Y.-D. Zhang, "Pathological brain detection based on AlexNet and transfer learning," Journal of Computational Science, vol. 30, pp. 41-47, Jan. 2019. [Online]. Available: https://doi.org/10.1016/j.jocs.2018.11.008.
[24] H.-C. Chen, A. M. Widodo, and A. Wisnujati, "AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf," Electronics, vol. 11, no. 6, pp. 951, Jun. 2022. [Online]. Available: https://doi.org/10.3390/electronics11060951.
[25] L. Zhu, Z. B. Li, C. Li, J. Wu, and J. Yue, "High performance vegetable classification from images based on AlexNet deep learning model," Int. J. Agric. Biol. Eng., vol. 11, no. 4, pp. 217-223, 2018. [Online]. Available: https://doi.org/10.25165/ijabe.v11n4.3050.
[26] S.-H. Wang, S. Xie, X. Chen, D. S. Guttery, C. Tang, J. Sun, and Y.-D. Zhang, "Alcoholism Identification Based on an AlexNet Transfer Learning Model," Frontiers in Psychiatry, vol. 10, 2019. [Online]. Available: https://doi.org/10.3389/fpsyt.2019.00205.
[27] F. Anowar, S. Sadaoui, and B. Selim, "Conceptual and Empirical Comparison of Dimensionality Reduction Algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)," Computer Science Review, vol. 40, May 2021, Article 100378. [Online]. Available: https://doi.org/10.1016/j.cosrev.2021.100378.
[28] H. Naeem and A. A. Bin-Salem, "A CNN-LSTM network with multi-level feature extraction-based approach for automated detection of coronavirus from CT scan and X-ray images," Applied Soft Computing, vol. 113, Part A, pp. 107918, Dec. 2021. [Online]. Available: https://doi.org/10.1016/j.asoc.2021.107918.
[29] P. Panda, O. K. C. U, S. Marappan, S. Ma, M. S, and D. V. Nandi, "Transfer Learning for Image-Based Malware Detection for IoT," Sensors, vol. 23, no. 6, p. 3253, 2023. [Online]. Available: https://doi.org/10.3390/s23063253.
[30] M. Almahmoud, D. Alzu'bi, and Q. Yaseen, "ReDroidDet: Android Malware Detection Based on Recurrent Neural Network," Procedia Computer Science, vol. 184, pp. 841-846, 2021. https://doi.org/10.1016/j.procs.2021.03.105.
[31] M. A. Khan, "HCRNNIDS: Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System," Processes, vol. 9, no. 5, p. 834, May 2021. [Online]. Available: https://doi.org/10.3390/pr9050834.