Turkish Journal of Computer and Mathematics Education *Vol.14 No.02 (2023),50- 63*
Research Article

INTELLIGENT MALWARE DETECTION USING EXTREME LEARNING MACHINE

V. Keerthi Sree[1], P. Shravani[1], V.Sravani[1], P. Devendar[2]

[1,2]Department of Information Technology

[1,2]Malla Reddy Engineering College for Women (A), Maisammaguda, Medchal, Telangana.

ABSTRACT

Now-a-days to detect cyber-attack are using static and dynamic analysis of request data. Static analysis is based on signature which we will match existing attack signature with new request packet data to identify packet is normal or contains attack signature. Dynamic analysis will use dynamic execution of program to detect malware/attack, but dynamic analysis is time consuming. To overcome from this problem and to increase detection accuracy with old and new malware attacks, we are using machine learning algorithms and evaluating prediction performance of various machine learning algorithms such as Support Vector Machine (SVM), Random Forest, Decision Tree, Naïve Bayes, Logistic Regression, KNearest Neighbours and Deep Learning Algorithms such as Convolution Neural Networks (CNN) and LSTM (Long Short-Term Memory). Among those, various models Deep learning CNN resulted in superior performance compared to other models. To implement this work and to evaluate machine learning algorithms performance this work using binary malware dataset called 'MALIMG'. This dataset contains 25 families of malware and application will convert this binary dataset into gray images to generate train and test models for machine learning algorithms. This algorithm converting binary data to images and then generating model, so they are called as MalConv CNN and MalConv LSTM and another algorithm refers as EMBER. Application convert dataset into binary images and then used 80% dataset for training model and 20% dataset for testing. Whenever we upload new test malware binary data then application will apply new test data on train model to predict malware class.

Keywords: Malware detection, extreme learning machine, cyber-attacks.

1. INTRODUCTION

In this digital world of Industry 4.0, the rapid advancement of technologies has affected the daily activities in businesses as well as in personal lives. Internet of Things (IoT) and applications have led to the development of the modern concept of the information society. However, security concerns pose a major challenge in realising the benefits of this industrial revolution as cyber criminal's attack individual PC's and networks for stealing confidential data for financial gains and causing denial of service to systems. Such attackers make use of malicious software or malware to cause serious threats and vulnerability of systems [1]. A malware is a computer program with the purpose of causing harm to the operating system (OS). A malware gets different names such as adware, spyware, virus, worm, trojan, rootkit, backdoor, ransomware and command and control (C&C) bot, based on its purpose and behaviour. Detection and mitigation of malware is an evolving problem in the cyber security field. As researchers develop new techniques, malware authors improve their ability to evade detection.

When Morris worm made its appearance as the first ever computer virus in 1988-89, antivirus software programs were designed to detect the existence of such a malware by finding a match with the virus definition database updated from time to time. This is called signature-based malware detection, which can also perform a heuristic search to identify the behavior of malware. However, the major challenge in such classical approaches is that new variants of malware use antivirus evasion techniques such as code obfuscation and hence such signature-based approaches are unable to detect zero-day malwares [2]. Signature-based malware detection system requires extensive domain level knowledge to reverse engineer the malware using Static and dynamic analysis and to assign a signature for that. Moreover, signature-based system requires larger time to reverse engineer the malware and during that time an attacker would encroach into the system. In addition, signature-based system fails to detect new types of malware. Security researchers have identified that hackers predominantly use polymorphism and metamorphism as obfuscation techniques against signature-based detection. In order to address this problem, software tools are used to manually unpack the codes and analyse the application programming interface (API) calls.

Since this process is a resource intensive task, [3] presented an automated system to extract API calls and analyse the malicious characteristics using a four step methodology. In step 1, the malware is unpacked. In step 2, the binary executable is disassembled. Step 3 involves API call extraction. Step 4 involves API call mapping and statistical feature analysis. This was enhanced in [4] using a 5- step methodology incorporating machine learning algorithm (MLA) such as SVM with n-gram features extracted from large samples of both the benign and malicious executables with 10-fold cross validations. Later, in [5] a comparative study of various classical machine learning classifiers for malware detection was performed, and a framework for zero day malware detection was proposed. To handle malicious code variants, the sequence of API calls and their frequency of appearance of API calls passed into similarity based mining and machine learning methods [7]. The detailed experimental analysis was done on very large data set and to extract the features from malware binaries a unified framework proposed. In [8], API calls features and a hybrid of support vector machine (SVM) and Maximum-Relevance Minimum Redundancy Filter (MRMRF) heuristics were employed to present novel feature selection approaches for enhanced malware detection. Recently, with the increase in unknown malware attacks, the detailed information on obfuscated malware is discussed by [6] and many researchers are improving the MLAs for malware detection [9]. This forms the motivation of this research work.

2. LITERATURE SURVEY

Machine learning algorithms (MLAs) rely on the feature engineering, feature selection and feature representation methods. The set of features with a corresponding class is used to train a model in order to create a separating plane between the benign and malwares. This separating plane helps to detect a malware and categorize it into its corresponding malware family. Both feature engineering and feature selection methods require domain level knowledge. The various features can be obtained through Static and Dynamic analysis. Static analysis is a method that captures the information from the binary program without executing. Dynamic analysis is the process of monitoring malware behavior at run time in an isolated environment. The complexities and various issues of Dynamic analysis are discussed in detail by [10]. Dynamic analysis can be an efficient long-term solution for malware detection system. The Dynamic analysis cannot be deployed in end-point real time malware detection due to the reason that it takes much time to analyze its behaviour, during which malicious payload can get delivered. Malware detection methods based on Dynamic analysis are more robust to obfuscation methods when compared to statically collected data. Most commonly, the commercial anti-malware solutions use a hybrid of Static and Dynamic analysis approaches. The major issue with the classical machine learning based malware detection system is that they rely on the feature engineering, feature learning and feature representation techniques that require an extensive domain level knowledge [11], [12], [13]. Moreover, once an attacker comes to know the features, the malware detector can be evaded easily [14]. To be successful, MLAs require data with a variety of patterns of malware. The publicly available benchmark data for malware analysis research is very less due to the security and privacy concerns. Though few datasets exist, each of them has their own harsh criticisms as most of them are outdated. Many of the published results of machine learning based malware analysis have used their own datasets. Even though publicly available sources exist to crawl the malware datasets, preparing a proper dataset for research is a daunting task. These issues are the main drawbacks behind developing generic machine learning based malware analysis system that can be deployed in real time. More importantly, the compelling issues in applying data science techniques were discussed in detail by [15].

In recent days, deep learning, which is an improved model of neural networks has outperformed the classical MLAs in many of the tasks which exist in the field of natural language processing (NLP), computer vision, speech processing and many others [16]. During the training process, it tries to capture higher level representation of features in deep hidden layers with the ability to learn from mistakes. MLAs experience diminishing outputs as they see more and more data whereas deep learning captures new patterns and establishes associations with the already captured pattern to enhance the performance of tasks. There exists few research studies towards the application of deep learning architectures for malware analysis to improve cyber security [13], [11], [12], [17], [18], [18]–[24]. However, with Industry 4.0, the number of malwares is rapidly increasing in recent times. Since the continuous collection of malwares in real time results in Big Data, the existing approaches are not scalable with very high requirements for storage and time in making efficient decisions.

3. PROPOSED METHODOLOGY

Deep learning or deep neural networks (DNNs) takes inspiration from how the brain works and forms a sub module of artificial intelligence. The main strength of deep learning architectures is the capability to understand the meaning of data when it is in large amounts and to automatically tune the derived meaning with new data without the need for a domain expert knowledge. Convolutional neural networks (CNNs) and Recurrent neural networks (RNNs) are two types of deep learning architectures predominantly applied in real-life scenarios. Generally, CNN architectures are used for spatial data and RNN architectures are used for temporal data. The combination of CNN and LSTM is used for spatial and temporal data analysis.

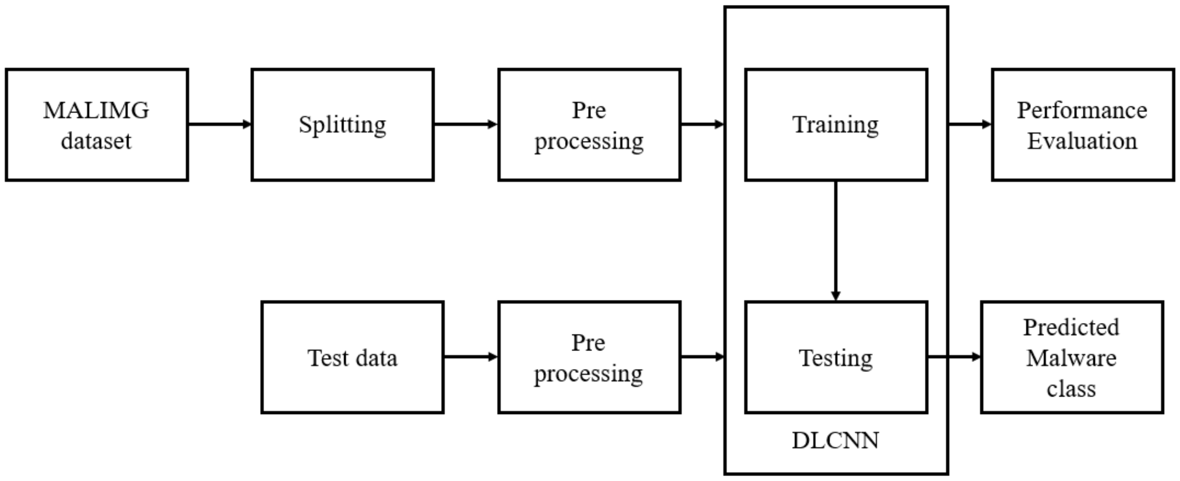


Fig. 1: Proposed block diagram.

Fig. 1 shows the block diagram of proposed method. Initially, MALIMG dataset is spitted into 80% for training and 20% for testing. Then, dataset preprocessing operation is performed to normalize the entire dataset. Further, DLCNN classifier is used for prediction of malware attack from test sample. The performance evaluation is carried out to show supremacy of proposed method.

3.1 MALIMG dataset

CICDDoS2019 contains benign and the most up-to-date common DDoS attacks, which resembles the true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter-V3 with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols and attack (CSV files). Generating realistic background traffic was our top priority in building this dataset. We have used our proposed B-Profile system to profile the abstract behaviour of human interactions and generates naturalistic benign background traffic in the proposed testbed. For this dataset, we built the abstract behaviour of 25 users based on the HTTP, HTTPS, FTP, SSH and email protocols.

3.2 DLCNN

A feed forward neural network (FFN) creates a directed graph in which a graph is composed of nodes and edges. FFN passes information along edges from one node to another without formation of a cycle. Multi-layer perceptron (MLP) is a type of FFN that contains 3 or more layers, specifically one input layer, one or more hidden layer and an output layer in which each layer has many neurons, called as units in mathematical notation. The number of hidden layers is selected by following a hyper parameter tuning approach. The information is transformed from one layer to another layer in forward direction without considering the past values. Moreover, neurons in each layer are fully connected.

Convolutional network or convolutional neural network or CNN is supplement to the classical feed forward network (FFN), primarily used in the field of data processing. Here, m denotes number of filters, ln denotes number of input features and p denotes reduced feature dimension, it depends on pooling length. In this work, CNN network composed of convolution 1D layer, pooling 1D layer and fully connected layer. A CNN network can have more than one convolution 1D layer, pooling 1D layer and fully connected layer. In convolutional 1D layer, the filters slide over the 1D sequence data and extracts optimal features. The features that are extracted from each filter are grouped into a new feature set called as feature map. The number of filters and the length are chosen by following a hyper parameter tuning method. This in turn uses non-linear activation function, ReLU on each element.

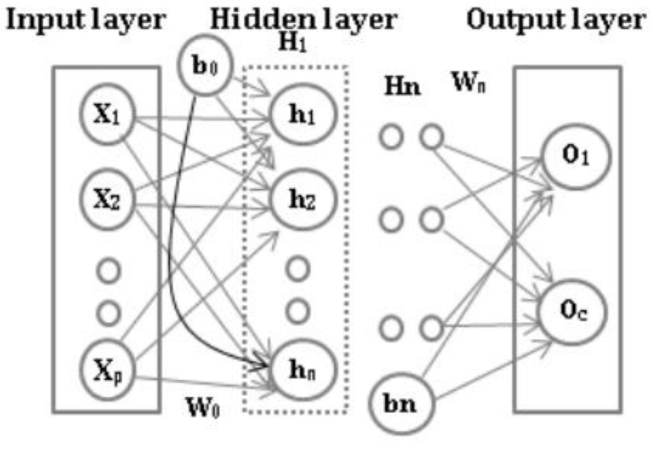


Fig. 2: DNN architecture.

The dimensions of the optimal features are reduced using pooling 1D layer using either max pooling, min pooling or average pooling. Since the maximum output within a selected region is selected in max pooling, we adopt max pooling in this work. Finally, the CNN network contains fully connected layer for classification. In fully connected layer, each neuron contains a connection to every other neuron. Instead of passing the pooling 1D layer features into fully connected layer, it can also be given to recurrent layer, LSTM to capture the sequence related information. Finally, the LSTM features are passed into fully connected layer for classification.

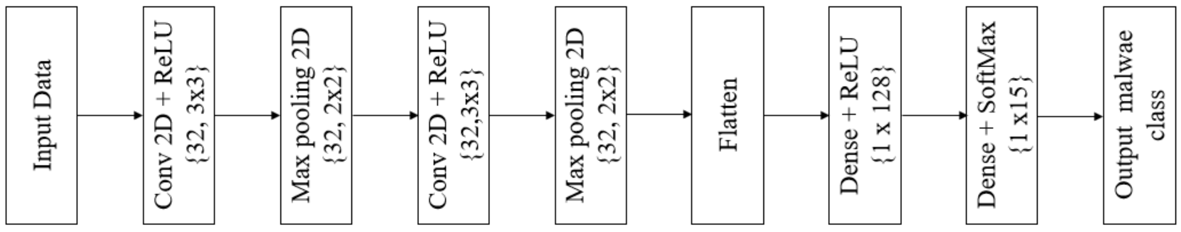


Fig. 3: Architecture of DLCNN for malware detection.

Table. 1: Layers description.

Layer Names	No. of filters	Kernel size	Feature size
Conv 2D +ReLU	32	3 x 3	62x62x32
Max pooling 2D	-	3 x 3	31x31x32
Conv 2D+ReLU	32	3 x 3	29x29x32
Max pooling 2D	-	3 x 3	14x14x32
Flatten	-	1x6272	1x6272
Dense +ReLU		1 x 128	1 x 256
Dense + SoftMax		1 x 15	1 x 15

Convolutional neural networks are generally composed of three parts. Convolution layer for feature extraction. The convergence layer, also known as the pooling layer, is mainly used for feature selection. The number of parameters is reduced by reducing the number of features. The full connection layer carries out the summary and output of the characteristics. A convolution layer is consisting of a convolution process and a nonlinear activation function ReLU. A typical architecture of CNN model for malware class recognition.

The leftmost data is the input layer, which the computer understands as the input of several matrices. Next is the convolution layer, the activation function of which uses ReLU. The pooling layer has no activation function. The combination of convolution and pooling layers can be constructed many times. The combination of convolution layer and convolution layer or convolution layer and pool layer can be very flexibly, which is not limited when constructing the model. But the most common CNN is a combination of several convolution layers and pooling layers. Finally, there is a full connection layer, which acts as a classifier and maps the learned feature representation to the sample label space.

Convolutional neural network mainly solves the following two problems.

1) Problem of too many parameters: It is assumed that the size of the input picture is $50 * 50 * 3$. If placed in a fully connected feedforward network, there are 7500 mutually independent links to the hidden layer. And each link also corresponds to its unique weight parameter. With the increase of the number of layers, the size of the parameters also increases significantly. On the one hand, it will easily lead to the occurrence of over-fitting phenomenon. On the other hand, the neural network is too complex, which will seriously affect the training efficiency. In convolutional neural networks, the parameter sharing mechanism makes the same parameters used in multiple functions of a model, and each element of the convolutional kernel will act on a specific position of each local input. The neural network only needs to learn a set of parameters and does not need to optimize learning for each parameter of each position.

2) Data stability: Data stability is the local invariant feature, which means that the natural data will not be affected by the scaling, translation, and rotation of the data size. Because in deep learning, data enhancement is generally needed to improve performance, and fully connected feedforward neural is difficult to ensure the local invariance of the data. This problem can be solved by convolution operation in convolutional neural network.

4. RESULTS AND DISCUSSION

To implement this project and to evaluate machine learning algorithms performance author is using binary malware dataset called ‘MALIMG’. This dataset contains 25 families of malware and application will convert this binary dataset into gray images to generate train and test models for machine learning algorithms. These algorithms converting binary data to images and then generating model, so they are called as MalConv CNN and MalConv LSTM and other algorithm refers as EMBER. Application convert dataset into binary images and then used 80% dataset for training model and 20% dataset for testing. Whenever we upload new test malware binary data then application will apply new test data on train model to predict malware class. In dataset total 25 families of malware, we can see and below are their names.

'Dialer Adialer.C','Backdoor Agent.FYI','Worm Allaple.A','Worm Allaple.L','Trojan Alueron.gen','Worm:AutoIT Autorun.K',

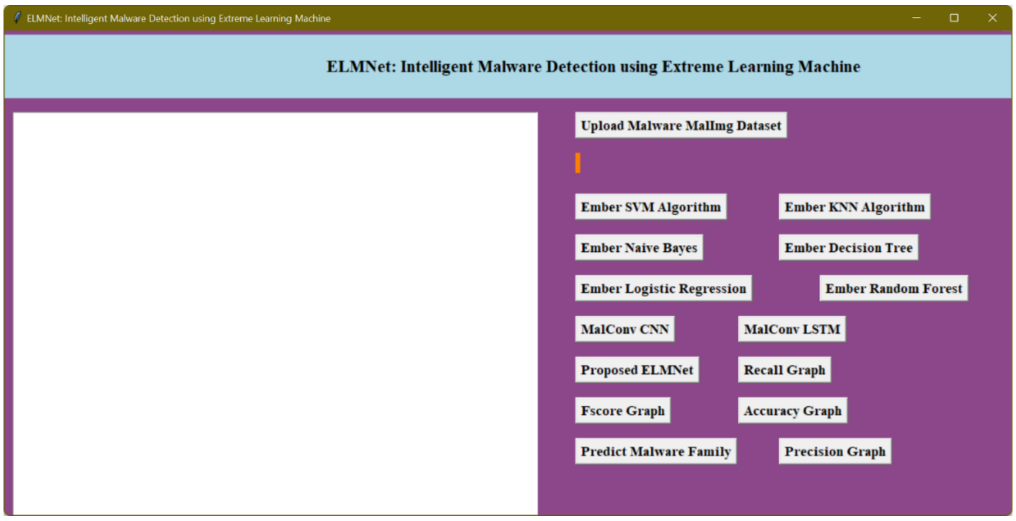
'Trojan C2Lop.P','Trojan C2Lop.gen','Dialer Dialplatform.B','Trojan Downloader Dontovo.A','Rogue Fakerean','Dialer Instantaccess',

'PWS Lolyda.AA 1','PWS Lolyda.AA 2','PWS Lolyda.AA 3','PWS Lolyda.AT','Trojan Malex.gen','Trojan Downloader Obfuscator.AD',

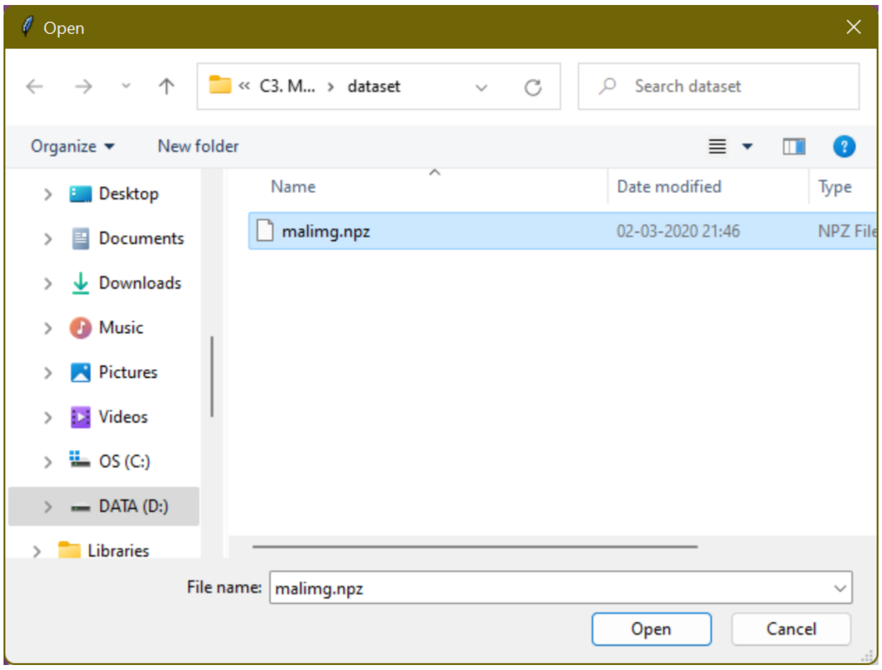
'Backdoor Rbot!gen','Trojan Skintrim.N','Trojan Downloader Swizzor.gen!E','Trojan Downloader Swizzor.gen!I','Worm VB.AT', 'Trojan Downloader Wintrim.BX','Worm Yuner.A'

All above names are the malware families

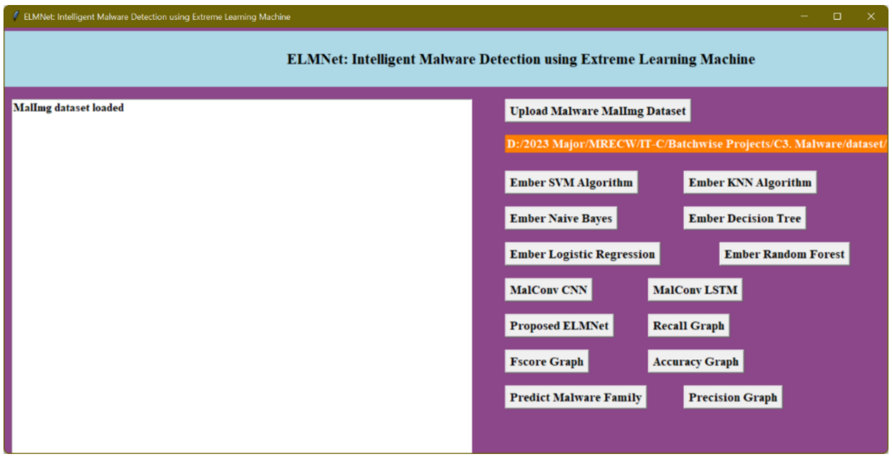
All new malware test files I saved inside images folder, and you can upload this files to predict it class. All algorithms detail you can read from paper.



In above screen click on ‘Upload Malware MalImg dataset’ button to upload dataset.

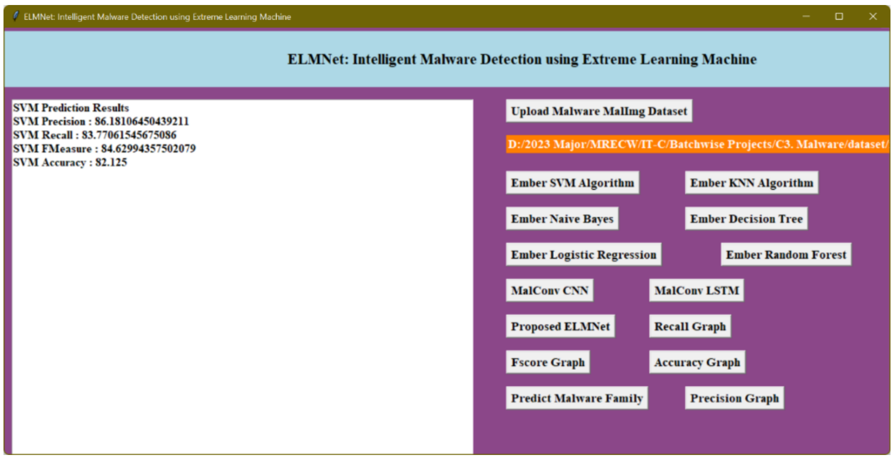


In above screen I am uploading ‘malimg.npz’ binary malware dataset and after uploading dataset will get below screen.


ELMNet: Intelligent Malware Detection using Extreme Learning Machine
ELMNet: Intelligent Malware Detection using Extreme Learning Machine
Upload Malware MalImg Dataset
D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/dataset/
Ember SVM Algorithm
Ember KNN Algorithm
Ember Naive Bayes
Ember Decision Tree
Ember Logistic Regression
Ember Random Forest
MalConv CNN
MalConv LSTM
Proposed ELMNet
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family
Precision Graph

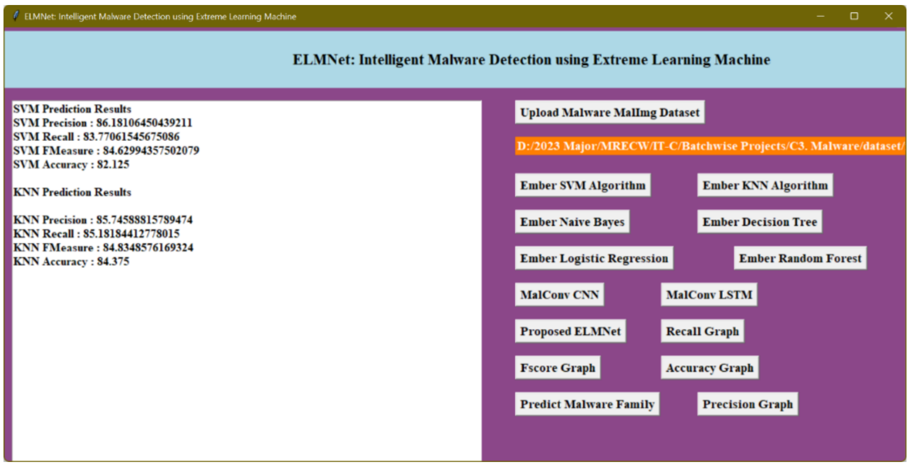
```
MalImg dataset loaded
```

Now click on ‘Ember SVM algorithm’ button to read malware dataset and generate train and test model and then apply SVM algorithm to calculate its prediction accuracy, FSCORE, Precision and Recall. If algorithm performance is good then its accuracy, precision or recall values will be closer to 100.


ELMNet: Intelligent Malware Detection using Extreme Learning Machine
ELMNet: Intelligent Malware Detection using Extreme Learning Machine
Upload Malware MalImg Dataset
D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/dataset/
Ember SVM Algorithm
Ember KNN Algorithm
Ember Naive Bayes
Ember Decision Tree
Ember Logistic Regression
Ember Random Forest
MalConv CNN
MalConv LSTM
Proposed ELMNet
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family
Precision Graph

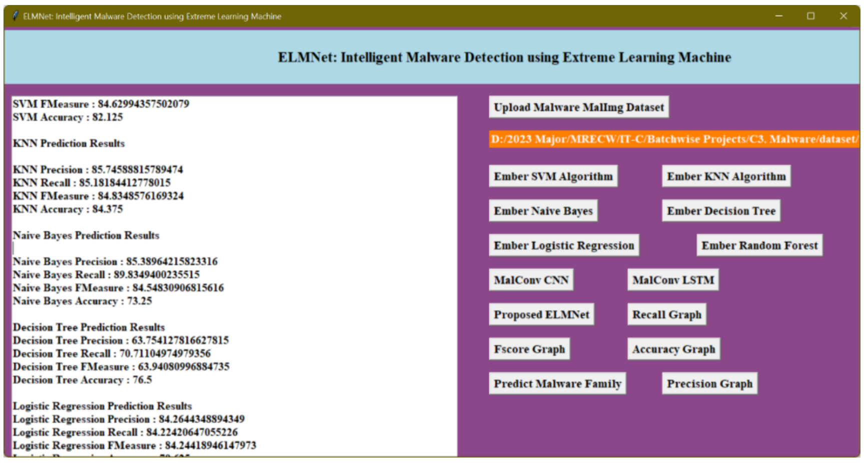
```
SVM Prediction Results
SVM Precision : 86.18106450439211
SVM Recall : 83.77061545675086
SVM FMeasure : 84.62994357502079
SVM Accuracy : 82.125
```

In above screen we got SVM precision, recall and F-Measure. Now click on ‘Ember KNN Algorithm’ button to get its performance


ELMNet: Intelligent Malware Detection using Extreme Learning Machine
ELMNet: Intelligent Malware Detection using Extreme Learning Machine
Upload Malware MalImg Dataset
D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/dataset/
Ember SVM Algorithm
Ember KNN Algorithm
Ember Naive Bayes
Ember Decision Tree
Ember Logistic Regression
Ember Random Forest
MalConv CNN
MalConv LSTM
Proposed ELMNet
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family
Precision Graph

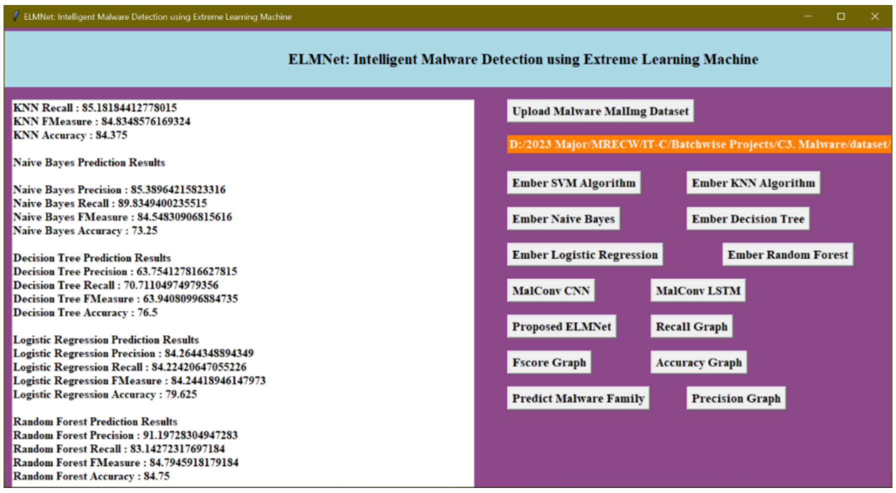
```
SVM Prediction Results
SVM Precision : 86.18106450439211
SVM Recall : 83.77061545675086
SVM FMeasure : 84.62994357502079
SVM Accuracy : 82.125

KNN Prediction Results

KNN Precision : 85.74588815789474
KNN Recall : 85.18184412778015
KNN FMeasure : 84.8348576169324
KNN Accuracy : 84.375
```

In above screen we got KNN details and now click on ‘Naïve Bayes’, Decision Tree and Logistic Regression buttons to get its performance details


ELMNet: Intelligent Malware Detection using Extreme Learning Machine
ELMNet: Intelligent Malware Detection using Extreme Learning Machine
Upload Malware MalImg Dataset
D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/dataset/
Ember SVM Algorithm
Ember KNN Algorithm
Ember Naive Bayes
Ember Decision Tree
Ember Logistic Regression
Ember Random Forest
MalConv CNN
MalConv LSTM
Proposed ELMNet
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family
Precision Graph

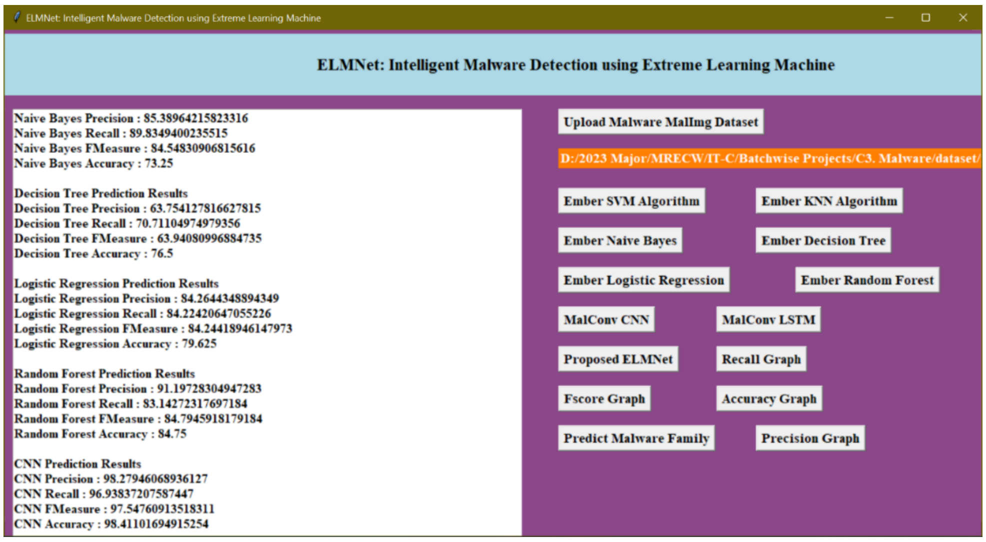
```
SVM FMeasure : 84.62994357502079
SVM Accuracy : 82.125

KNN Prediction Results

KNN Precision : 85.74588815789474
KNN Recall : 85.18184412778015
KNN FMeasure : 84.8348576169324
KNN Accuracy : 84.375

Naive Bayes Prediction Results

Naive Bayes Precision : 85.38964215823316
Naive Bayes Recall : 89.8349400235515
Naive Bayes FMeasure : 84.54830906815616
Naive Bayes Accuracy : 73.25

Decision Tree Prediction Results
Decision Tree Precision : 63.754127816627815
Decision Tree Recall : 70.71104974979356
Decision Tree FMeasure : 63.94080996884735
Decision Tree Accuracy : 76.5

Logistic Regression Prediction Results
Logistic Regression Precision : 84.2644348894349
Logistic Regression Recall : 84.22420647055226
Logistic Regression FMeasure : 84.24418946147973
```

In above screen we got Naïve Bayes, Decision Tree and logistic regression details and now click on ‘Random Forest’ button to get its performance


ELMNet: Intelligent Malware Detection using Extreme Learning Machine
ELMNet: Intelligent Malware Detection using Extreme Learning Machine
Upload Malware MalImg Dataset
D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/dataset/
Ember SVM Algorithm
Ember KNN Algorithm
Ember Naive Bayes
Ember Decision Tree
Ember Logistic Regression
Ember Random Forest
MalConv CNN
MalConv LSTM
Proposed ELMNet
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family
Precision Graph

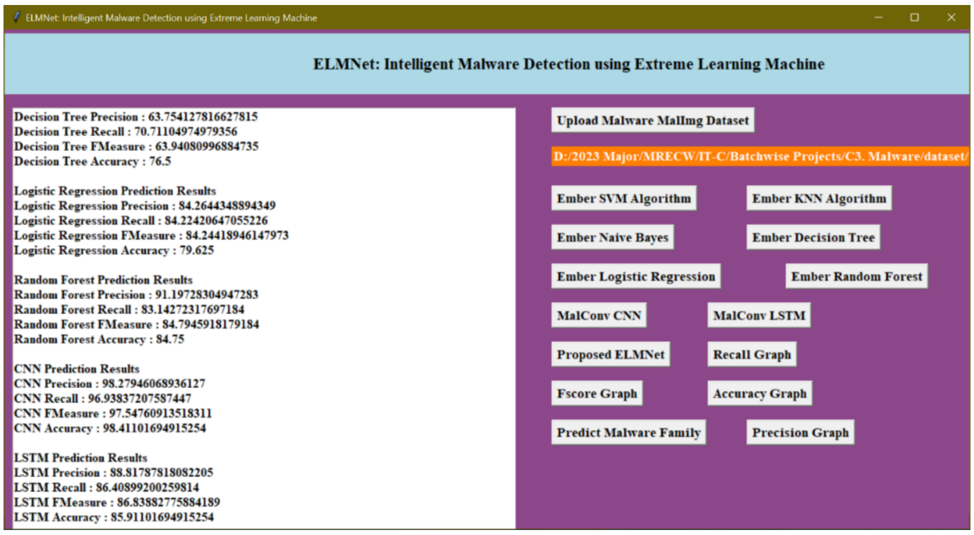
```
KNN Recall : 85.18184412778015
KNN FMeasure : 84.8348576169324
KNN Accuracy : 84.375

Naive Bayes Prediction Results

Naive Bayes Precision : 85.38964215823316
Naive Bayes Recall : 89.8349400235515
Naive Bayes FMeasure : 84.54830906815616
Naive Bayes Accuracy : 73.25

Decision Tree Prediction Results
Decision Tree Precision : 63.754127816627815
Decision Tree Recall : 70.71104974979356
Decision Tree FMeasure : 63.94080996884735
Decision Tree Accuracy : 76.5

Logistic Regression Prediction Results
Logistic Regression Precision : 84.2644348894349
Logistic Regression Recall : 84.22420647055226
Logistic Regression FMeasure : 84.24418946147973
Logistic Regression Accuracy : 79.625

Random Forest Prediction Results
Random Forest Precision : 91.19728304947283
Random Forest Recall : 83.14272317697184
Random Forest FMeasure : 84.7945918179184
Random Forest Accuracy : 84.75
```

In above screen we got random forest details and now click on ‘MalConv CNN’ button to get its performance details.


ELMNet: Intelligent Malware Detection using Extreme Learning Machine
ELMNet: Intelligent Malware Detection using Extreme Learning Machine
Upload Malware MalImg Dataset
D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/dataset/
Ember SVM Algorithm
Ember KNN Algorithm
Ember Naive Bayes
Ember Decision Tree
Ember Logistic Regression
Ember Random Forest
MalConv CNN
MalConv LSTM
Proposed ELMNet
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family
Precision Graph

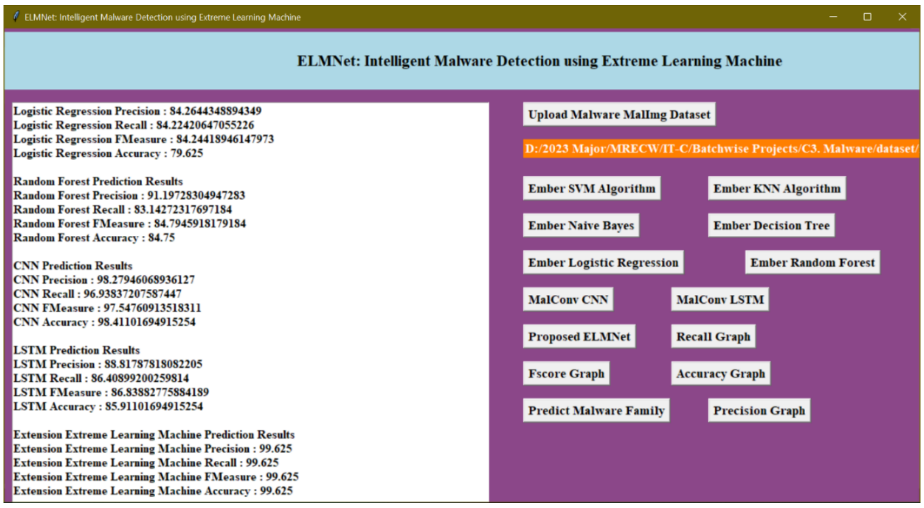
```
Naive Bayes Precision : 85.38964215823316
Naive Bayes Recall : 89.8349400235515
Naive Bayes FMeasure : 84.54830906815616
Naive Bayes Accuracy : 73.25

Decision Tree Prediction Results
Decision Tree Precision : 63.754127816627815
Decision Tree Recall : 70.71104974979356
Decision Tree FMeasure : 63.94080996884735
Decision Tree Accuracy : 76.5

Logistic Regression Prediction Results
Logistic Regression Precision : 84.2644348894349
Logistic Regression Recall : 84.22420647055226
Logistic Regression FMeasure : 84.24418946147973
Logistic Regression Accuracy : 79.625

Random Forest Prediction Results
Random Forest Precision : 91.19728304947283
Random Forest Recall : 83.14272317697184
Random Forest FMeasure : 84.7945918179184
Random Forest Accuracy : 84.75

CNN Prediction Results
CNN Precision : 98.27946068936127
CNN Recall : 96.93837207587447
CNN FMeasure : 97.54760913518311
CNN Accuracy : 98.41101694915254
```

In above screen we got CNN performance values and now click on ‘MalConv LSTM’ button to run LSTM algorithm.


ELMNet: Intelligent Malware Detection using Extreme Learning Machine
ELMNet: Intelligent Malware Detection using Extreme Learning Machine
Upload Malware MalImg Dataset
D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/dataset/
Ember SVM Algorithm
Ember KNN Algorithm
Ember Naive Bayes
Ember Decision Tree
Ember Logistic Regression
Ember Random Forest
MalConv CNN
MalConv LSTM
Proposed ELMNet
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family
Precision Graph

```
Decision Tree Precision : 63.754127816627815
Decision Tree Recall : 70.71104974979356
Decision Tree FMeasure : 63.94080996884735
Decision Tree Accuracy : 76.5

Logistic Regression Prediction Results
Logistic Regression Precision : 84.2644348894349
Logistic Regression Recall : 84.22420647055226
Logistic Regression FMeasure : 84.24418946147973
Logistic Regression Accuracy : 79.625

Random Forest Prediction Results
Random Forest Precision : 91.19728304947283
Random Forest Recall : 83.14272317697184
Random Forest FMeasure : 84.7945918179184
Random Forest Accuracy : 84.75

CNN Prediction Results
CNN Precision : 98.27946068936127
CNN Recall : 96.93837207587447
CNN FMeasure : 97.54760913518311
CNN Accuracy : 98.41101694915254

LSTM Prediction Results
LSTM Precision : 88.81787818082205
LSTM Recall : 86.40899200259814
LSTM FMeasure : 86.83882775884189
LSTM Accuracy : 85.91101694915254
```

In above screen we can see LSTM details. In above screen we can see accuracy and other metrics from various algorithms where CNN got 98.41% accuracy which is higher as compared all the above algorithms. Now click on proposed ELMNet to get below output.


ELMNet: Intelligent Malware Detection using Extreme Learning Machine
ELMNet: Intelligent Malware Detection using Extreme Learning Machine
Upload Malware MalImg Dataset
D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/dataset/
Ember SVM Algorithm
Ember KNN Algorithm
Ember Naive Bayes
Ember Decision Tree
Ember Logistic Regression
Ember Random Forest
MalConv CNN
MalConv LSTM
Proposed ELMNet
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family
Precision Graph

```
Logistic Regression Precision : 84.2644348894349
Logistic Regression Recall : 84.22420647055226
Logistic Regression FMeasure : 84.24418946147973
Logistic Regression Accuracy : 79.625

Random Forest Prediction Results
Random Forest Precision : 91.19728304947283
Random Forest Recall : 83.14272317697184
Random Forest FMeasure : 84.7945918179184
Random Forest Accuracy : 84.75

CNN Prediction Results
CNN Precision : 98.27946068936127
CNN Recall : 96.93837207587447
CNN FMeasure : 97.54760913518311
CNN Accuracy : 98.41101694915254

LSTM Prediction Results
LSTM Precision : 88.81787818082205
LSTM Recall : 86.40899200259814
LSTM FMeasure : 86.83882775884189
LSTM Accuracy : 85.91101694915254

Extension Extreme Learning Machine Prediction Results
Extension Extreme Learning Machine Precision : 99.625
Extension Extreme Learning Machine Recall : 99.625
Extension Extreme Learning Machine FMeasure : 99.625
Extension Extreme Learning Machine Accuracy : 99.625
```

In above screen CNN got 98.41% accuracy and proposed ELMNet got 99.62% accuracy which is higher than any other algorithm. So, by employing ELM we can further increase malware prediction capability of the application. Now click on ‘Precision, Recall & F-Measure’ button to get comparison graph for all metrics and all algorithms

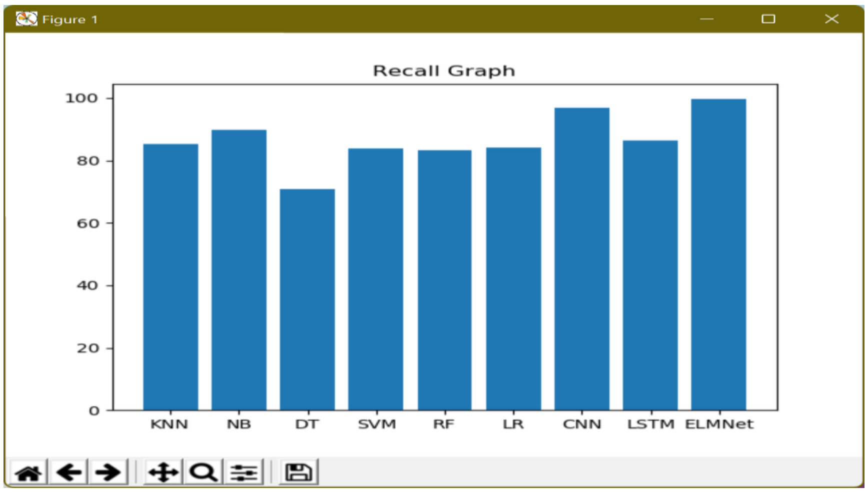


Figure 1
Recall Graph
100
80
60
40
20
0
KNN
NB
DT
SVM
RF
LR
CNN
LSTM
ELMNet

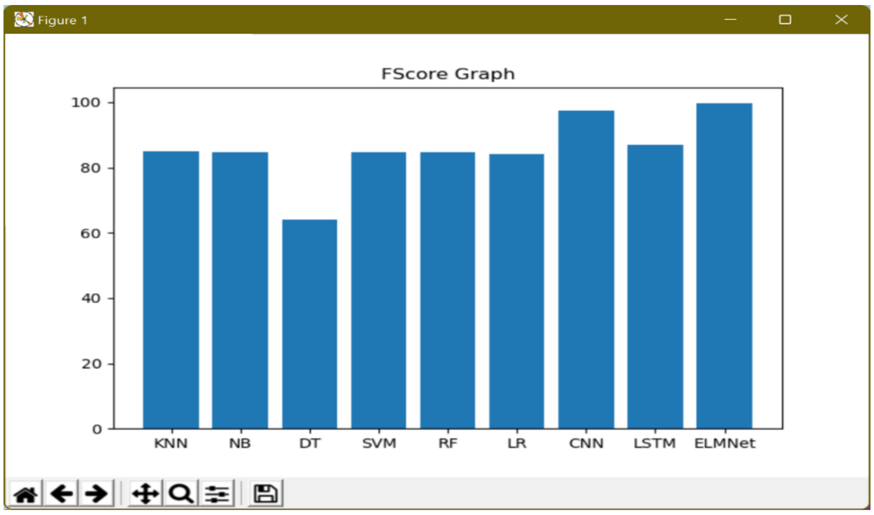


Figure 1
FScore Graph
100
80
60
40
20
0
KNN
NB
DT
SVM
RF
LR
CNN
LSTM
ELMNet

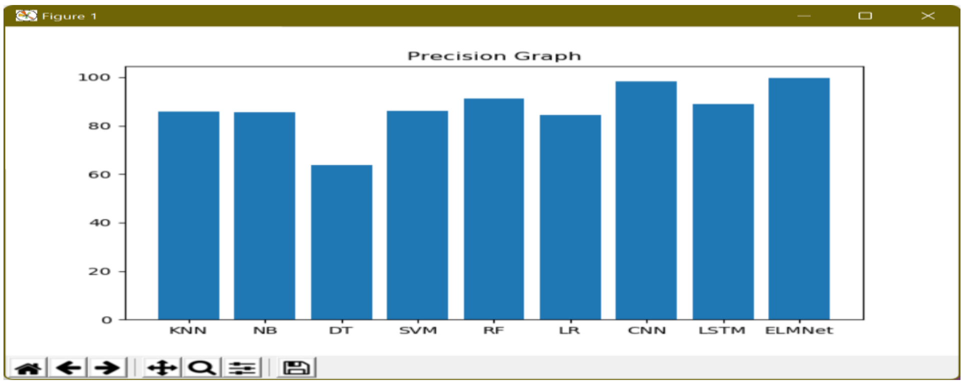
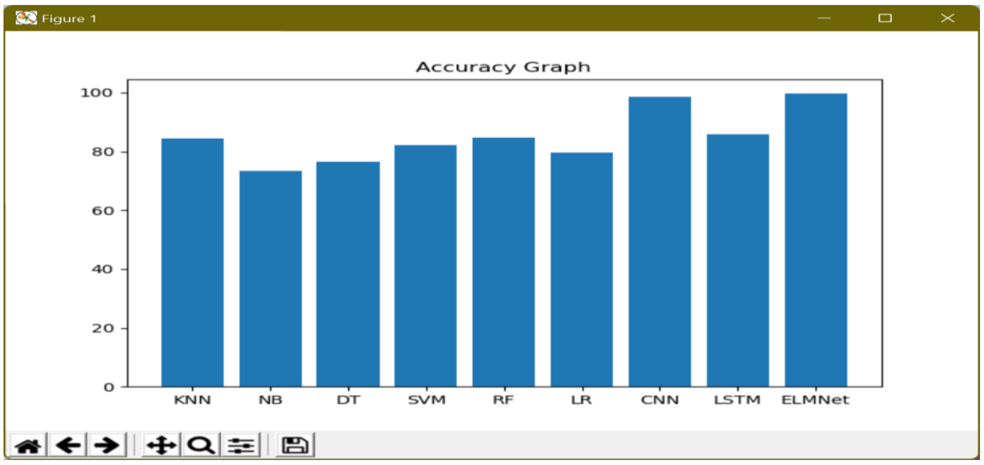
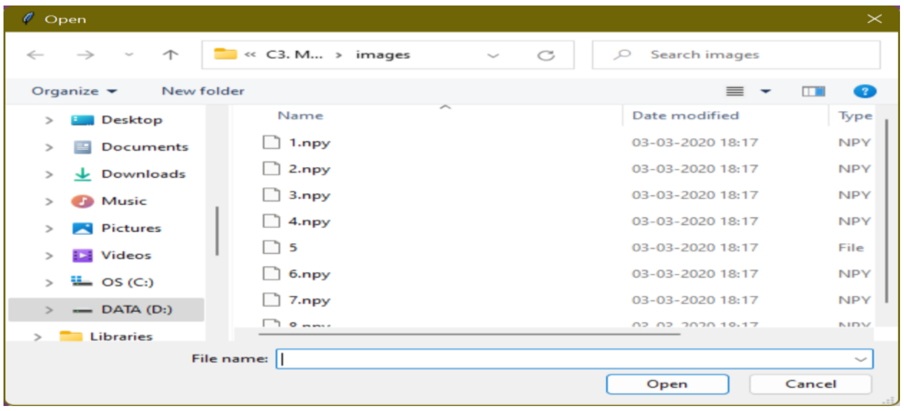


Figure 1
Precision Graph
100
80
60
40
20
0
KNN
NB
DT
SVM
RF
LR
CNN
LSTM
ELMNet

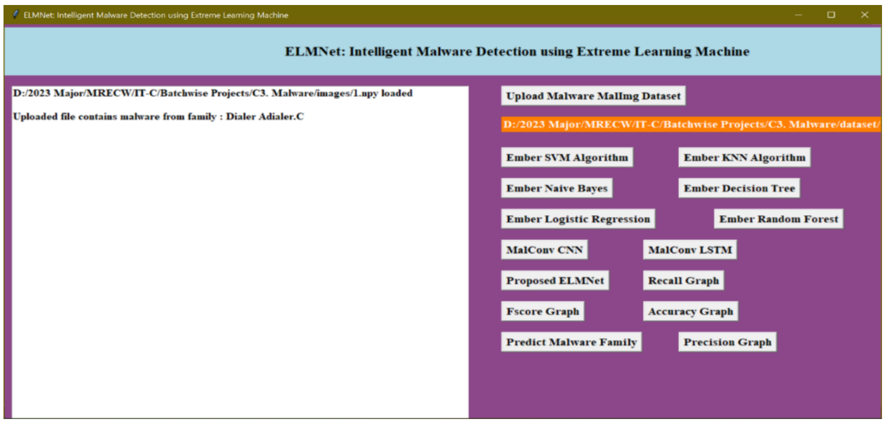


Now click on ‘Predict Malware Family’ button and upload binary file to get or predict class of malware.



Name	Date modified	Type
1.npy	03-03-2020 18:17	NPY
2.npy	03-03-2020 18:17	NPY
3.npy	03-03-2020 18:17	NPY
4.npy	03-03-2020 18:17	NPY
5	03-03-2020 18:17	File
6.npy	03-03-2020 18:17	NPY
7.npy	03-03-2020 18:17	NPY

In above graph I am uploading one binary file called 1.npy and below is the malware prediction of that file.



ELMNet: Intelligent Malware Detection using Extreme Learning Machine

D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/dataset/

```
D:/2023 Major/MRECW/IT-C/Batchwise Projects/C3. Malware/images/1.npy loaded

Uploaded file contains malware from family : Dialer Adialer.C
```

In above screen we can see uploaded test file contains ‘Dialer Adialer.C’ malware attack. Similarly, we can upload other files and predict class.

5. CONCLUSION AND FUTURE SCOPE

This project proposed an efficient malware detection and designed a highly scalable framework to detect, classify and categorize zero-day malwares. This framework applies neural network on the collected malwares from end user hosts and follows a two-stage process for malware analysis. In the first stage, a hybrid of static and dynamic analysis was applied for malware classification. In the second stage, malwares were grouped into corresponding malware categories using image processing approaches. Various experimental analysis conducted by applying variations in the models on publicly available benchmark dataset and indicated the proposed model outperformed classical MLAs. The developed framework is capable of analyzing large number of malwares in real-time and scaled out to analyse even larger number of malwares by stacking a few more layers to the existing architectures. Future research entails exploration of these variations with new features that could be added to the existing data.

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] M. Tang, M. Alazab, and Y. Luo, ‘‘Big data for cybersecurity: Vulnerability disclosure trends and dependencies,’’ IEEE Trans. Big Data, to be published.

[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] M. Alazab, S. Venkatraman, P. Watters, M. Alazab, and A. Alazab, ‘‘Cybercrime: The case of obfuscated malware,’’ in Global Security, Safety and Sustainability & e-Democracy (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 99, C. K. Georgiadis, H. Jahankhani, E. Pimenidis, R. Bashroush, and A. Al-Nemrat, Eds. Berlin, Germany: Springer, 2012.

[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] R. Verma, ''Security analytics: Adapting data science for security challenges,'' in Proc. 4th ACM Int. Workshop Secur. Privacy Anal. New York, NY, USA: ACM, Mar. 2018, pp. 40–41.

[16] Y. LeCun, Y. Bengio, and G. Hinton, ''Deep learning,'' Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[17] A. F. Agarap and F. J. H. Pepito. (2017). ''Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification.'' [Online]. Available: https://arxiv.org/abs/1801.00318

[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 Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Apr. 2015, pp. 1916–1920.

[23] T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, ''Efficient dynamic malware analysis based on network behavior using deep learning,'' in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2016, pp. 1–7.

[24] S. H. Ebenuwa, M. S. Sharif, M. Alazab, and A. Al-Nemrat, ''Variance ranking attributes selection techniques for binary classification problem in imbalance data,'' IEEE Access, vol. 7, pp. 24649–24666, 2019.

[25] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, ''Malware images: Visualization and automatic classification,'' in Proc. 8th Int. Symp. Vis. Cyber Secur. New York, NY, USA: ACM, Jul. 2011, p. 4.

[26] F. C. C. Garcia, and F. P. Muga, II, (2016). ''Random forest for malware classification.'' [Online]. Available: https://arxiv.org/abs/arXiv: 1609.07770

[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