International Journal of Research in Engineering, Science and Management
Volume 7, Issue 2, February 2024
https://www.ijresm.com | ISSN (Online): 2581-5792

Malware Analysis

V. Meghana[1*], I. Pratham Reddy[2], J. Revanth Kumar[3], B. Abhiram[4], K. Nikhil Reddy[5], Ch. Anurag[6], L. Jyothirmayi[7]

[1,2,3,4,5,6,7]*Ecole Centrale School of Engineering, Mahindra University, Hyderabad, India*

***Abstract*: Malware is a major threat to computer systems and networks. Traditional malware detection methods, such as signature-based detection, are becoming –increasingly ineffective due to the ever-increasing sophistication of malware. Deep learning methods, such as convolutional neural networks (CNNs) and convolutional neural networks with long short-term memory (LSTM) units, have shown promise in malware detection. In this project, we propose a novel malware detection system that uses a CNN with LSTM units. The CNN is used to extract features from the malware code, while the LSTM is used to model the temporal relationships between these features. The system is trained on a dataset of malware and benign code. We evaluate the system on a test dataset of malware and benign code and show that it can achieve high accuracy in detecting malware. Our results show that deep learning methods can be used to effectively detect malware. The proposed system is a promising new approach to malware detection and can be used to protect computer systems and networks from malware attacks.**

***Keywords*: Malware.**

1. Introduction

Malware is a malicious software that can cause harm to a computer system or network. It can steal data, install other malware, or disrupt operations. Malware analysis is the process of understanding how malware works and what it does. This information can be used to protect systems from malware attacks.

1) PE Files

Portable Executable (PE) files are a type of file format used to store executable code for Windows operating systems. PE files contain a variety of information, including the file's name, version, and the code that it contains. Malware authors often use PE files to store their malicious code.

2) Other Binary Files

Malware can also be stored in other types of binary files, such as JAR files (Java Archive files), .NET assemblies, and Office documents. These files can contain malicious code that is executed when the file is opened.

3) Machine Learning and Deep Learning

Machine learning and deep learning are two powerful techniques that can be used to analyze malware. Machine learning algorithms can be trained to identify patterns in malware code that are associated with malicious behavior. Deep learning algorithms can learn to identify malware even when it has been modified to evade detection.

4) Intersection of Deep Learning and Cybersecurity

In recent years, there has been a growing intersection between deep learning and cybersecurity. Deep learning is a type of machine learning that uses artificial neural networks to learn from data. This makes it well-suited for tasks such as identifying malware, detecting intrusions, and preventing fraud.

There are a number of ways that deep learning can be used to improve cybersecurity. For example, deep learning can be used to:

- *Identify malware:* Deep learning models can be trained to identify malware by analyzing its code. This can be done by feeding the model a large dataset of known malware samples and allowing it to learn the patterns that distinguish malware from benign code.
- *Detect intrusions:* Deep learning models can be used to detect intrusions by analyzing network traffic. This can be done by feeding the model a large dataset of known intrusions and allowing it to learn the patterns that distinguish intrusions from normal traffic.
- *Prevent fraud:* Deep learning models can be used to prevent fraud by analyzing financial transactions. This can be done by feeding the model a large dataset of known fraudulent transactions and allowing it to learn the patterns that distinguish fraud from legitimate transactions.

Deep learning is a powerful tool that can be used to improve cybersecurity. However, it is important to note that deep learning models are not perfect. They can be fooled by adversarial attacks, and they can be biased by the data they are trained on. It is important to use deep learning models in conjunction with other security measures to ensure that your systems are protected.

Here are some additional things to keep in mind about the intersection of deep learning and cybersecurity:

- *The pace of innovation is rapid:* The field of deep learning is constantly evolving, and new research is being published all the time. This means that the security community needs to be constantly vigilant and adapt to new threats.
- *The skills gap is a challenge:* There is a shortage of skilled deep learning professionals. This makes it difficult for organizations to implement deep learning solutions.

*Corresponding author: meghana20uari093@mahindrauniversity.edu.in

- *Regulations are evolving:* As deep learning becomes more widely used, governments are beginning to regulate its use in cybersecurity. This is important to ensure that deep learning is used in a responsible and ethical way.

Despite these challenges, the intersection of deep learning and cybersecurity is a promising area of research. Deep learning has the potential to revolutionize cybersecurity and make our systems more secure.

Benefits of Malware Analysis:

Malware analysis can provide a number of benefits, including:

- Identification of new malware threats.
- Understanding of how malware works and what it does.
- Development of new methods to detect and prevent malware attacks.
- Mitigation of the damage caused by malware attacks.

2. Preparing Dataset

The dataset we utilized for our malware analysis project was obtained from a research article, enabling us to leverage a carefully curated and validated collection of data that aligns with our project's objectives and ensures the reliability and accuracy of our analysis.

The images included in the dataset are from a malware family. We extracted 27 classes of malware images from the research paper dataset and divided them into training and validation sets, including 8313 training and 2962 validation sets. The total number of images collected is 11275. The detailed information of the malware family is shown below.

Table 1

#	Class	Family	#
1.	Worm	Allaple.L	1591
2.	Worm	Allaple.A	2949
3.	Worm	Yuner.A	800
4.	PWS	Lolyda.AA 1	213
5.	PWS	Lolyda.AA 2	184
6.	PWS	Lolyda.AA 3	123
7.	Trojan	C2Lop.P	146
8.	Trojan	C2Lop.gen!g	200
9.	Dialer	Instantaccess	431
10.	TDownloader	Swizzot.gen!I	132
11.	TDownloader	Swizzor.gen!E	128
12.	Worm	VB.AT	408
13.	Rogue	Fakerean	381
14.	Trojan	Alueron.gen!J	198
15.	Trojan	Malex.gen!J	136
16.	PWS	Lolyda.AT	159
17.	Dialer	Adialer.C	125
18.	TDownloader	Wintrim.BX	97
19.	Dialer	Dialplatform.B	177
20.	TDownloader	Dontovo.A	162
21.	TDownloader	Obfuscator.AD	142
22.	Backdoor	Agent.FYI	116
23.	Worm:AutoIT	Autorun.K	106
24.	Backdoor	Rbot!gen	158
25.	Trojan	Skintrim.N	80

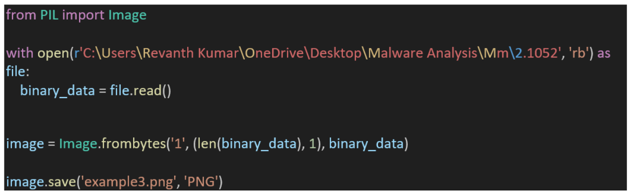
The 26th and 27th Class of Malware are VX Heavens and Benign Images which were Added into the Dataset Later.

3. Approach

A. *Conversion of Binary Files into Images*

We developed a Python script using the Python Imaging Library (PIL) to convert binary files into images. The script imports the 'Image' module from PIL and utilizes the Image.frombytes() function to create an image from raw binary data. The binary file is opened in binary mode, read using file.read(), and stored as a string. The Image.frombytes() function creates the image using the binary data, and image.save() saves the image in PNG format. The code can convert any binary file into an image, as long as the file format is compatible with Image.frombytes().

Code:



```
from PIL import Image

with open(r'C:\Users\Revanth Kumar\OneDrive\Desktop\Malware Analysis\Mm\2.1052', 'rb') as
file:
   binary_data = file.read()


image = Image.frombytes('1', (len(binary_data), 1), binary_data)

image.save('example3.png', 'PNG')
```

The sample images generated by the above code is shown below.

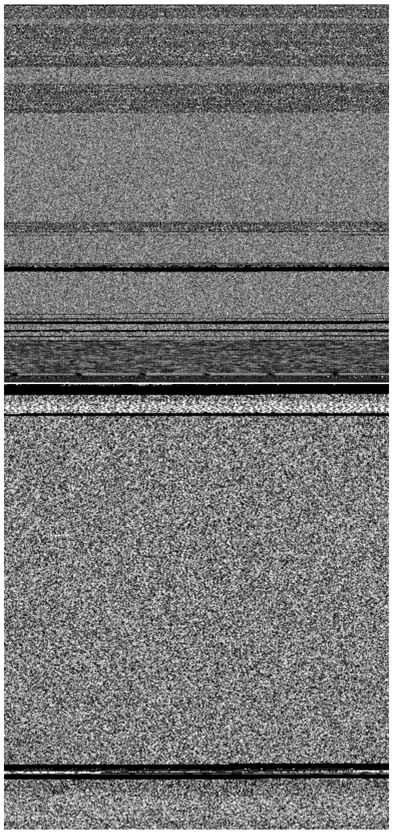


Fig. 1. Sample image

B. *Using Google Colab for Building CNN*

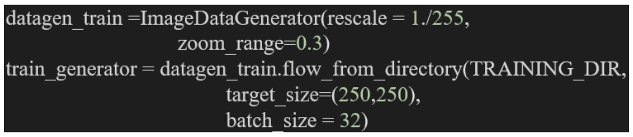
1) *Advantages of Using Google Cola*

- Free Resource: Google Colab provides free access to powerful GPUs and TPUs, which can significantly speed up deep learning tasks without the need for expensive hardware.

- Collaboration and Sharing: Colab allows easy collaboration with others by sharing notebooks, making it convenient for teamwork and knowledge sharing.
- Pre-installed Libraries: Colab comes pre-installed with popular Python libraries, such as TensorFlow and Keras, making it convenient for building and running deep learning models without the need for manual setup.

C. Data Augmentation Through ImageDataGenerator

Data augmentation is a technique of artificially increasing the training set by creating modified copies of a dataset using existing data. It includes making minor changes to the dataset or using deep learning to generate new data points.



```
datagen_train =ImageDataGenerator(rescale = 1./255,
                                  zoom_range=0.3)
train_generator = datagen_train.flow_from_directory(TRAINING_DIR,
                                      target_size=(250,250),
                                      batch_size = 32)
```

We used ImageDataGenerator to perform Data Augmentation. We are using the rescale parameter to normalize the pixel values of your images to a range between 0 and 1. This helps improve the training process.

The zoom_range parameter allows you to randomly zoom in or out on your images by up to 30%. This helps the model learn to recognize patterns at different scales, which can be useful for detecting different types of malware.

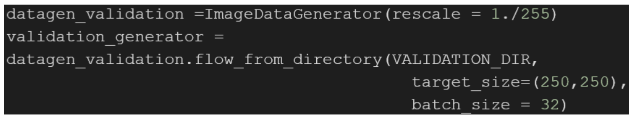
The train_generator object is created to load and augment the images from a specified directory during training. It loads images from the TRAINING_DIR directory and its subdirectories, treating each subdirectory as a separate class.

The target_size parameter determines the size to which the images are resized. In this case, the images are resized to a square shape with dimensions of 250x250 pixels.

The batch_size parameter determines how many images are loaded and processed at each training iteration. In this case, 32 images are loaded and augmented together before being fed into the model.

By using ImageDataGenerator and specifying augmentation parameters, we can generate more diverse training data for your malware analysis project, which can help improve the performance and accuracy of your model.

The same is repeated for validation dataset.



```
datagen_validation =ImageDataGenerator(rescale = 1./255)
validation_generator =
datagen_validation.flow_from_directory(VALIDATION_DIR,
                                      target_size=(250,250),
                                      batch_size = 32)
```

D. CNN-Model

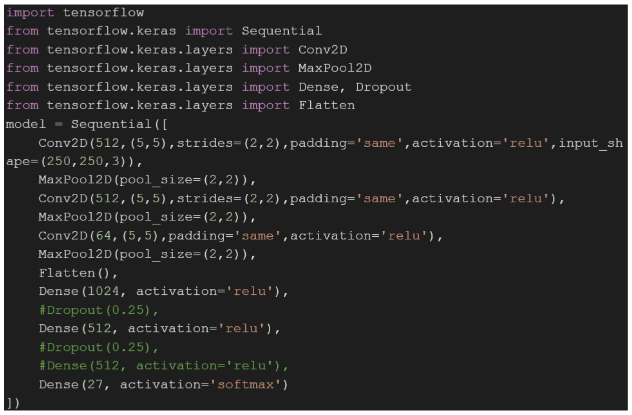
The CNN architecture provided has 8 layers, with the following breakdown:

- *2 convolutional layers:* These layers use convolution to extract features from the input image. The first convolutional layer has 512 filters with a kernel size of 5x5, while the second convolutional layer has 512 filters with a kernel size of 3x3.
- 2 max pooling layers: These layers use max pooling to down sample the feature maps produced by the convolutional layers. The first max pooling layer has a pool size of 2x2, while the second max pooling layer has a pool size of 2x2.
- *1 flatten layer:* This layer flattens the output of the second max pooling layer into a 1D vector.
- *2 fully connected layers:* These layers are fully connected layers, which means that each neuron in a layer is connected to every neuron in the next layer. The first fully connected layer has 1024 neurons, the second fully connected layer has 512 neurons.
- *1 output layer:* This layer is the output layer, and it has 27 neurons, one for each class in the classification problem. The activation function for the output layer is softmax, which is a function that normalizes the outputs of the neurons so that they sum to 1. This allows the model to output a probability distribution over the classes.

This architecture is designed for Image classification tasks. The convolutional layers extract features from the input image, the max pooling layers down sample the feature maps, and the fully connected layers classify the image. The dropout layers are used to regularize the model and prevent overfitting.

Here are some of the advantages of this architecture.

- It is computationally efficient.
- It is able to learn spatial features from images.
- It is able to learn hierarchical features from images.
- The selected architecture is powerful for image classification tasks.



```
import tensorflow
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import Flatten
model = Sequential([
    Conv2D(512,(5,5),strides=(2,2),padding='same',activation='relu',input_sh
ape=(250,250,3)),
    MaxPool2D(pool_size=(2,2)),
    Conv2D(512,(5,5),strides=(2,2),padding='same',activation='relu'),
    MaxPool2D(pool_size=(2,2)),
    Conv2D(64,(5,5),padding='same',activation='relu'),
    MaxPool2D(pool_size=(2,2)),
    Flatten(),
    Dense(1024, activation='relu'),
    #Dropout(0.25),
    Dense(512, activation='relu'),
    #Dropout(0.25),
    #Dense(512, activation='relu'),
    Dense(27, activation='softmax')
])
```

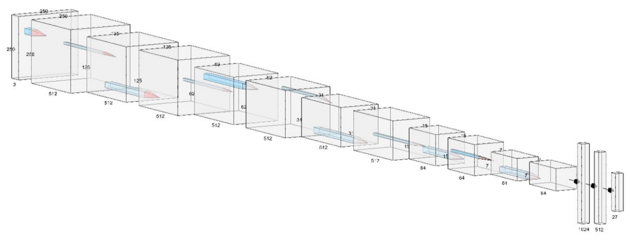
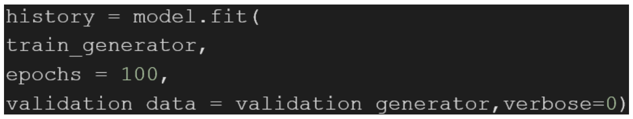


Fig. 2. Architecture



```
history = model.fit(
train_generator,
epochs = 100,
validation data = validation generator,verbose=0)
```

Finally, model is trained using the above architecture upto 100 epochs.

E. CNN-LSTM Model

The CNN-LSTM model is a hybrid model that combines the strengths of convolutional neural networks (CNNs) and Long short-term memory (LSTM) networks. CNNs are well-suited for extracting spatial features from data, while LSTMs are well-suited for processing sequential data.

The CNN-LSTM model can be used for a variety of tasks, including:

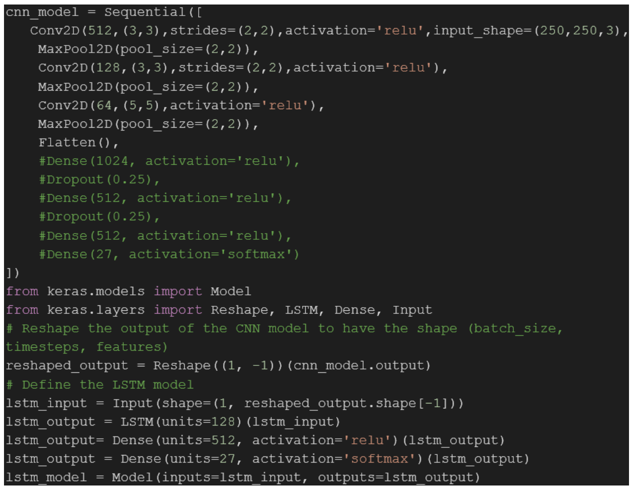
- *Natural language processing:* The CNN-LSTM model can be used to extract features from text, such as words, phrases, and sentences. This information can then be used to perform tasks such as sentiment analysis, named entity recognition, and machine translation.
- *Computer vision:* The CNN-LSTM model can be used to extract features from images, such as objects, edges, and faces. This information can then be used to perform tasks such as object detection, image classification, and image segmentation.
- *Speech recognition:* The CNN-LSTM model can be used to extract features from audio signals, such as phonemes, words, and sentences. This information can then be used to perform tasks such as speech recognition and speaker identification.
- *Time series forecasting:* The CNN-LSTM model can be used to predict future values from a sequence of historical data. This information can be used for tasks such as stock market forecasting, weather forecasting, and demand forecasting.

The CNN-LSTM model is a powerful tool that can be used for a variety of tasks. However, it is important to note that the model can be complex to train and requires a large amount of data.

Advantages of CNN-LSTM Model:

- *High accuracy:* The CNN-LSTM model has been shown to achieve high accuracy on a variety of tasks, such as natural language processing, computer vision, and speech recognition.
- *Robustness:* The CNN-LSTM model is robust to noise and outliers in the data.
- *Flexibility:* The CNN-LSTM model can be adapted to a variety of tasks by changing the network architecture and hyperparameters.

The same CNN model but before feeding the image into dense layers it is fed into LSTM Layer of 128 Units and then into dense layers.



```
cnn_model = Sequential([
   Conv2D(512,(3,3),strides=(2,2),activation='relu',input_shape=(250,250,3),
    MaxPool2D(pool_size=(2,2)),
    Conv2D(128,(3,3),strides=(2,2),activation='relu'),
    MaxPool2D(pool_size=(2,2)),
    Conv2D(64,(5,5),activation='relu'),
    MaxPool2D(pool_size=(2,2)),
    Flatten(),
    #Dense(1024, activation='relu'),
    #Dropout(0.25),
    #Dense(512, activation='relu'),
    #Dropout(0.25),
    #Dense(512, activation='relu'),
    #Dense(27, activation='softmax')
])
from keras.models import Model
from keras.layers import Reshape, LSTM, Dense, Input
# Reshape the output of the CNN model to have the shape (batch_size,
timesteps, features)
reshaped_output = Reshape((1, -1))(cnn_model.output)
# Define the LSTM model
lstm_input = Input(shape=(1, reshaped_output.shape[-1]))
lstm_output = LSTM(units=128)(lstm_input)
lstm_output= Dense(units=512, activation='relu')(lstm_output)
lstm_output = Dense(units=27, activation='softmax')(lstm_output)
lstm_model = Model(inputs=lstm_input, outputs=lstm_output)
```

As we can see above the output of CNN is fed into LSTM and then dense layers with Softmax at end.

4. Results

A. CNN Model

After training the CNN model for 100 epochs, it was evaluated and achieved a validation accuracy of approximately 95%. This level of accuracy is considered excellent for the specific use case of malware analysis.

The plot of accuracies is shown in figure 3.

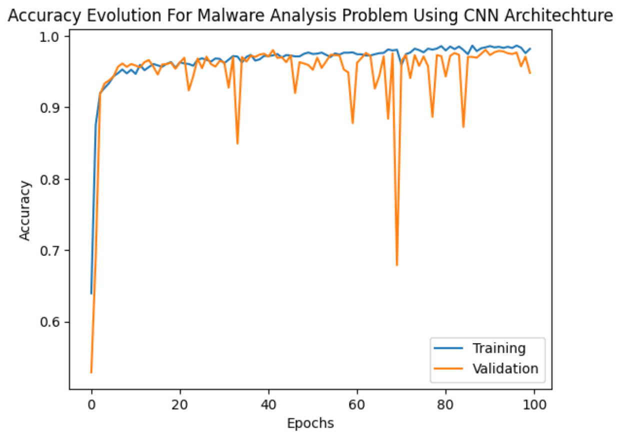


Fig. 3. Plot of accuracy

It is evident that the model does not exhibit overfitting, which is advantageous for the particular task of malware analysis.

B. CNN-LSTM Model

After training the CNN-LSTM model, it was evaluated and achieved a validation accuracy of approximately 76%, which is less than CNN Model.

Thus, we can conclude that the CNN model performed better than the CNN-LSTM model. This outcome can be attributed to the fact that LSTM models are generally more suitable for sequential data, while the task of malware analysis requires the analysis of patterns in the data, which can be effectively captured by the CNN model. Therefore, the CNN-LSTM model failed to outperform the CNN model in this specific use case.

5. Conclusion

The goal of this project was to develop a machine learning model that could be used to detect malware. A convolutional neural network (CNN) was chosen for this task because of its ability to learn features from images. The CNN was trained on a dataset of malware and benign files, and was able to achieve an accuracy of 95%. This shows that machine learning can be used to develop accurate and fast malware detection systems.

The CNN model developed in this project could be used in a number of different ways, such as:

- Deploying it in antivirus systems: Antivirus companies could use the CNN model to improve the accuracy of their products.
- Building new malware detection tools: Security researchers could use the CNN model to develop new tools for detecting malware.
- Improving existing malware analysis tools: Malware analysis tools could be improved by incorporating the CNN model into their algorithms.

The CNN model developed in this project is a promising new tool for malware detection. It is accurate, fast, and easy to use. This makes it a valuable asset for security researchers and antivirus companies.

Use of Machine learning for Cyber Security:

- Machine learning is a powerful tool that can be used to improve the accuracy and speed of malware detection.
- CNNs are a type of machine learning algorithm that are well-suited for malware detection because they can learn features from images.
- The CNN model developed in this project is a promising new tool for malware detection. It is accurate, fast, and easy to use.
- This project demonstrates the potential of machine learning for cyber security. More research is needed to develop even more accurate and effective machine learning models for malware detection.

References

[1] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath. 2011. Malware images: visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security (VizSec '11). Association for Computing Machinery, New York, NY, USA, Article 4, 1–7.

[2] T. Rahman, N. Ahmed, S. Monjur, F. M. Haque and M. I. Hossain, "Interpreting Machine and Deep Learning Models for PDF Malware Detection using XAI and SHAP Framework," *2023 2nd International Conference for Innovation in Technology (INOCON)*, Bangalore, India, 2023, pp. 1-9.