National University
of computer and emerging sciences

FYP-I Project Report

Semi-Supervised Learning based IoT Malware Classifier

***Author*:**		***Supervisor:***
Huzaifa Shahid	17k-3794	Ms. Anam Qureshi
Saad Amin	17k-3850	***Co-Supervisor:***
Azhan Ali	17k-3775	Dr. Jawwad Shamsi

Department of Computer Science, Karachi

December 14, 2020

December 2020

National University of Computer & Emerging Sciences

Supervisor	Ms. Anam Qureshi
CoSupervisor	Dr. Jawwad Shamsi
Member 1	Huzaifa Shahid
Member 2	Saad Amin
Member 3	Azhan Ali
Submission date	December 14, 2020

Supervisor

Ms. Anam Qureshi ________________________

Co-Supervisor

Sir Jawwad Shamsi ________________________

Department of Computer Science

National University of Computer & Emerging Sciences,

Main Campus, Karachi

ABSTRACT

We propose a semi-supervised deep learning framework for malware classification and labeling. Malware creators have successfully been able to bypass the traditional non-machine learning approach to distinguish malware with benign code by simply adding or removing an unusable part of code, hence, changing the complete signature. Recent deep learning approaches have been successful in capturing the underlying patterns in images, such as Convolutional Neural Networks (CNN). To capture the underlying patterns, we will convert our malware binaries to gray scale images and train a CNN for classification of different classes of malware. As we know in this modern era data is very important especially labeled data which is not easily available. In order to produce efficient test results more labeled data is required for training. But unfortunately a large amount of labeled data is unavailable or too expensive. To solve this constraint we propose a semi-supervised deep learning approach in our research project through which we will be able to train our model using unlabeled data by using semi-supervised learning algorithms such as constraint clustering and others. Lastly, a comparison analysis will be provided to compare the results of different semi-supervised learning approaches.

CHAPTERS

1 INTRODUCTION………………………………………………………………………… 5

1.1 Need for Solution……………………………………………………………………

1.2 About Semi supervised Approach…………………………………………………

1.3 Gap Analysis with Existing Solution………………………………………………

2 LITERATURE REVIEW………….………………………………….......................... 6

3 PROPOSED WORK ……….....……………………………………………………......12

3.1 Process Diagram…………………………………………………………......

3.2 Preprocessing……………………………………………………………….. 13

3.3 Deep Learning Classification Model…………………………….….… 14

3.4 Validation…………………………………………………………….….… 15

3.5 Testing…………………………………………………………………….…..

3.5 Semi-Supervised Learning Approach………………………………… 16

4 EXPERIMENTAL SETUP……………………………………………………………......19

4.1 Hardware ....

4.2 Software...

4.3 Dataset.. 20

5 RESULT……………………………………………………………………………......... 21

6 RESULT DESCRIPTION 22

7 CONCLUSION AND FUTURE WORK... 23

8 REFERENCES……… …………………………………………………….…… 24

1. Introduction

1.1 Need for Solution

The concept of malware security has taken huge steps in recent years to prevent devices being vulnerable to attacks. Malware is categorized into different types based on their functions like computer worm, computer viruses, Trojan etc. There has been a significant increase in the malware in recent years which poses potential security threat to organizations, businesses and individuals. To control the mutation of such harmful programs, fast and up-to-date systems need to be developed to quickly classify whether a file is malicious in nature or not.

1.2 About Semi-Supervised Approach

To produce efficient malware classification models we use CNN with semi supervised learning. Semi-supervised learning is a learning paradigm concerned with the study of how computers and natural systems such as humans learn in the presence of both labeled and unlabeled data [1].

1.3 GAP Analysis with Existing Solutions

Before semi-supervised learning training has been done either using an unsupervised approach in which all the data is unlabeled or a supervised approach in which all the data is labeled. The semi supervised learning uses both i.e. supervised and unsupervised approaches and produce better results. The main goal of semi supervised learning is to use less label data and utilize unlabeled data with it. As the label data is limited and expensive so utilizing unlabeled data with it makes the model more efficient. Today semi supervised learning is used in many applications. Few of them are speech analysis, internet content classification, and protein sequence classification. Today there are many state of the art algorithms that are present in semi supervised learning such as Pseudo-Labeling, Graph based learning, mixture models etc. These semi-supervised algorithms have shown good performance along with CNN on different image data-sets i.e., SHVN, NORB, MINIST, CIFAR100 etc. In our malware classification problem we use a semi supervised CNN model which helps us classify these malware with less label data more efficiently.

2. Literature Review

Deep Learning by Yann leCun, Yoshua Bengio, Geoffrey Hinton (2016) [5]

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the back propagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

CNN by Saad Albawi, Tareq Abed Mohammed (2017) [6]

In this paper they discussed the important issues related to Convolutional Neural Network (CNN) and explained the effect each parameter has on network performance. The most important layer in CNN is the convolution layer which takes most of the time within the network. Network performance also depends on the number of levels within the network. But on the other hand as the number of levels increases the times required to train and test the network. Today CNN is considered a powerful tool within machine learning for a lot of applications such as face detection and image, video recognitions and voice recognition.

Ensemble Learning by Omer Sagi, Lior Rokach (2017) [7]

Ensemble methods imitate human nature by seeking several opinions before making an important decision. The main idea of such methods is to weigh several individual models, and combine them in order to improve predictive performance. This paper has reviewed the major approaches and techniques in the field while discussing the core principles of training ensembles and combining their inducers' outputs. There are several trends and future research directions of ensemble learning that have been discussed in this paper. One trend is to refine popular algorithms to be more efficient and suitable for "Big Data". These efforts are usually involved in enabling the distribution of algorithms across multiple machines and the improvement of computational resource utilization. Another promising research direction is the transformation of ensemble models into models that are simpler and more comprehensive while preserving the predictive accuracy of the ensemble they were sourced from. Finally, they presented a recent study that aims at integrating the ensemble paradigm with deep neural networks, in accordance with the increasing pace of deep neural network research.

Malware Detection using Sequence Classification Methods and Ensembles by Jake Drew, Michael Hashler and Tyler Moore (2017) [8]

In this paper they have demonstrated how modern gene sequence classification tools can be applied to large-scale malware detection. In this first study, we have shown how the gene sequence classifier Strand can be used to predict multiple classes of polymorphic malware using data provided by the Kaggle Microsoft Malware Classification Challenge (Big 2015). While the approach, using only minimal adaptation, did not best the accuracy scores achieved by the highly tailored approach that won the competition, they did achieve classification accuracy levels exceeding 98% while making predictions over seven times faster than the training times required by the winning team. From the success of this demonstration, it is concluded that gene sequence classifiers in general and Strand in particular, hold great promise in their application to security datasets. In addition to polymorphic malware, they have anticipated that these classifiers can be used anywhere data sequences are used, such as in network traces of attacks or the identification of ransomware.

Regularization with Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning by Mehdi Sajjadi, Mehran Javanmardi and Tolga Tasdizen (2016) [2]

This research paper introduces unsupervised cost functions which remove non-deterministic behavior from the model and shows the advantage offered by this unsupervised cost function on using semi supervised learning with convolutional neural networks. This research paper mainly focuses on semi supervised deep learning. There has always been interest in exploiting unlabeled data to improve the performance of CNN. Few approaches are discussed in paper one approach is to use unlabeled data to pre-train the filters of CNN. The goal is to reduce the number of epochs required to converge and improve the accuracy compared to a model trained by random initialization [2]. This research paper introduces the loss function that minimizes the mean squared differences between different passes of an individual training sample through the network. When we use these loss functions we get good accuracy as compared to other cost functions because it removes randomness from the model. So that's why this loss function is also beneficial for semi supervised learning because in semi supervised learning we have less labelled data. This research paper introduces two unsupervised loss functions namely "**Transformation/stability cost function**" and "**Mutual-Exclusive loss function**". Research paper shows that on (MNIST DATASET) by using the combination of these two unsupervised cost functions we get almost the same error rate on 100 labelled data which is 0.55 and by using the whole data set we get 0.27 [2]. Similarly on CIFAR10 and CIFAR100 the model shows similar results. The state of the art error rate on the CIFAR100 dataset is 23.82% [2]. Combination of both unsupervised loss functions improves the accuracy even further and shows an error rate of 21.43% [2]. This cost function also shows good results on SHVN, NORB, and Image Net dataset as well.

A Cluster-then-label Semi Supervised Learning Approach for Pathology Image Classification by Mohammad Peikari, Sherine Salama, Sharon Nofech-Mozes & Anne L. Marte (2018) [9]

In the real world completely labeled datasets are hard to obtain. Semi-supervised learning methods are able to learn from both labeled and unlabeled data points. In this paper the underlying structure of the data space was investigated using cluster analysis. A cluster-then-label method was introduced to cluster high-density regions and the resulting cluster is used to label unlabeled data points after learning a classifier on labeled instances in each cluster.

A Semi-Supervised convolutional neural network-based method for steel surface defect recognition by Gao Yiping, Gao Liang, Li Xinyu, Yan Xuguo (2019) [3]

This paper focuses on automatic steel defect detection by using CNN (Convolutional Neural Network) with semi-supervised learning. To train a good model a large amount of data is required so that the model detects the underlying pattern present in the data. But unfortunately large amounts of labeled data is expensive and limited. With a limited dataset model unable to identify the underlying pattern. The semi-supervised learning, using both labeled and unlabeled samples for model training, can overcome this problem well. In this paper, a semi-supervised convolutional neural net-work (CNN) is proposed for steel surface defect recognition. The proposed method requires less labeled data and uses unlabeled data along with labeled data for training. The semi supervised algorithm used in this proposed method is Pseudo-Label. Pseudo-Label is the simple semi-supervised algorithm with efficient results. The results on a benchmark dataset of steel surface defect recognition shows that this method gives efficient results with limited labeled data, which achieves an accuracy of 90.7% with 17.53% improvement [3].

Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning by Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, Kevin McGuinness (2020) [4]

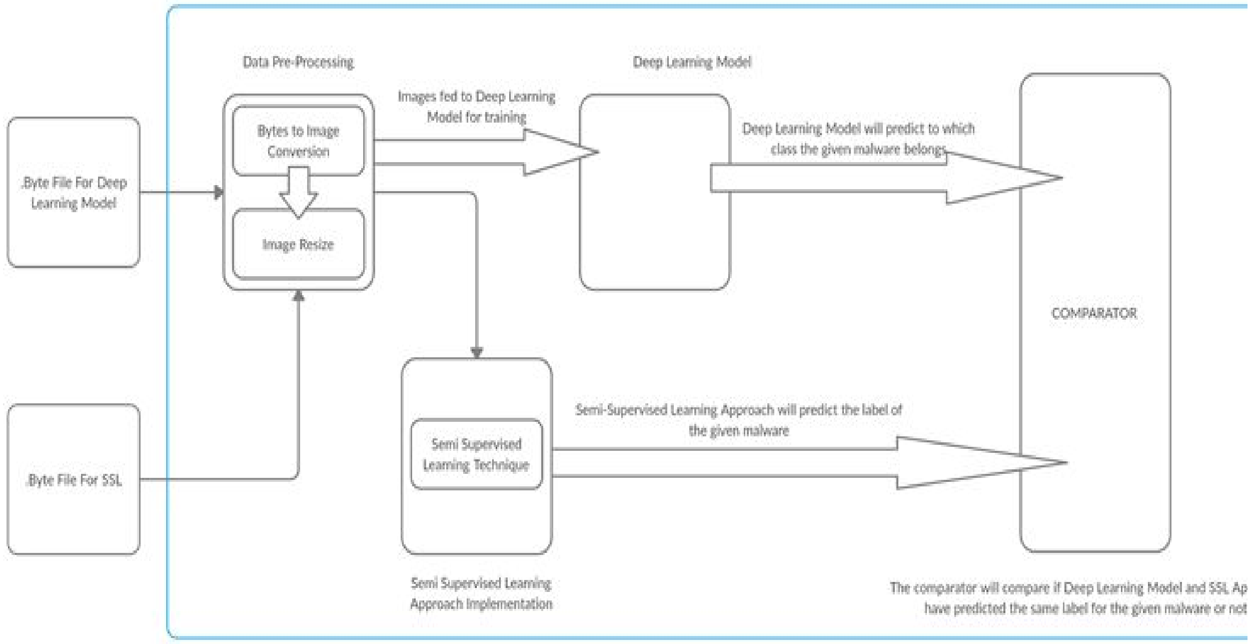
In this research paper semi supervised learning is used to train models on both labeled and unlabeled image dataset. This paper proposes to learn from unlabeled data by generating soft pseudo-labels. This paper shows that pseudo-labeling overfits to incorrect pseudo-labels due to the so-called confirmation bias and also that setting up a minimum number of labelled data per mini-batch is an effective regularization technique. The proposed method directly uses the network predictions as soft pseudo-labels for unlabeled data together with mix-up augmentation, a minimum number of labeled samples per mini-batch, dropout and data augmentation to alleviate confirmation bias [4]. The proposed method shows state of the art accuracy on dataset like CIFAR-10/100, SVHN, and Mini-ImageNet.

The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks by Dong-Hyun Lee[11]

All of the successful methods for training deep neural networks have something in common: they rely on an unsupervised learning algorithm [12] Most work in two main phases. In a first phase, unsupervised pre-training, the weights of all layers are initialized by layer-wise unsupervised training[9]. In a second phase, fine-tuning, the weights are trained globally with labels using backpropagation algorithm in a supervised fashion[11]. In this paper the author proposed a semi supervised approach i.e. pseudo labelling which is the same as **ENTROPY REGULARIZATION** (regularization technique in reinforcement learning) which is used in the fine tuning phase. In this paper deep neural network train in two phases in first phase the unsupervise algorithm used for pretraining is denoising autoencoder (DAE) and for fine tuning pseudo labelling is used. The proposed method gives best results on MNIST dataset with limited labels. With only 3000 labelled images the error rate is 2.69[11]

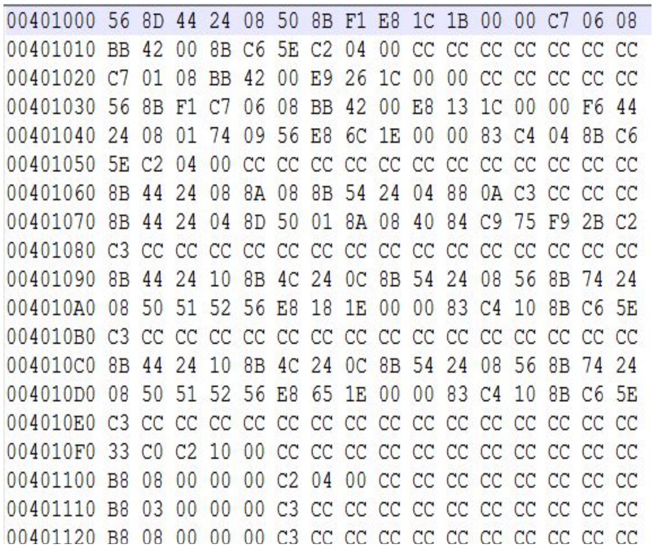
3. PROPOSED WORK

3.1 Process Diagram


Data Pre-Processing
Deep Learning Model
.Byte File For Deep Learning Model
Bytes to Image Conversion
Image Resize
Images fed to Deep Learning Model for training
Deep Learning Model will predict to which class the given malware belongs
COMPARATOR
.Byte File For SSL
Semi Supervised Learning Technique
Semi-Supervised Learning Approach will predict the label of the given malware
Semi Supervised Learning Approach Implementation
The comparator will compare if Deep Learning Model and SSL Ap
have predicted the same label for the given malware or not

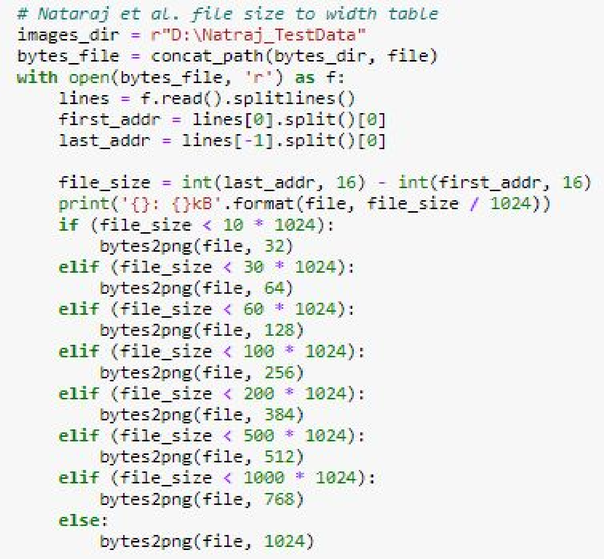
3.2 Preprocessing

Fetching Bytes File: We fetch only bytes file from dataset each bytes file contain bytes in hexadecimal format



```
00401000 56 8D 44 24 08 50 8B F1 E8 1C 1B 00 00 C7 06 08
00401010 BB 42 00 8B C6 5E C2 04 00 CC CC CC CC CC CC CC
00401020 C7 01 08 BB 42 00 E9 26 1C 00 00 CC CC CC CC CC
00401030 56 8B F1 C7 06 08 BB 42 00 E8 13 1C 00 00 F6 44
00401040 24 08 01 74 09 56 E8 6C 1E 00 00 83 C4 04 8B C6
00401050 5E C2 04 00 CC CC CC CC CC CC CC CC CC CC CC CC
00401060 8B 44 24 08 8A 08 8B 54 24 04 88 0A C3 CC CC CC
00401070 8B 44 24 04 8D 50 01 8A 08 40 84 C9 75 F9 2B C2
00401080 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
00401090 8B 44 24 10 8B 4C 24 0C 8B 54 24 08 56 8B 74 24
004010A0 08 50 51 52 56 E8 18 1E 00 00 83 C4 10 8B C6 5E
004010B0 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
004010C0 8B 44 24 10 8B 4C 24 0C 8B 54 24 08 56 8B 74 24
004010D0 08 50 51 52 56 E8 65 1E 00 00 83 C4 10 8B C6 5E
004010E0 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
004010F0 33 C0 C2 10 00 CC CC CC CC CC CC CC CC CC CC CC
00401100 B8 08 00 00 00 C2 04 00 CC CC CC CC CC CC CC CC
00401110 B8 03 00 00 00 C3 CC CC CC CC CC CC CC CC CC CC
00401120 B8 08 00 00 00 C3 CC CC CC CC CC CC CC CC CC CC
```

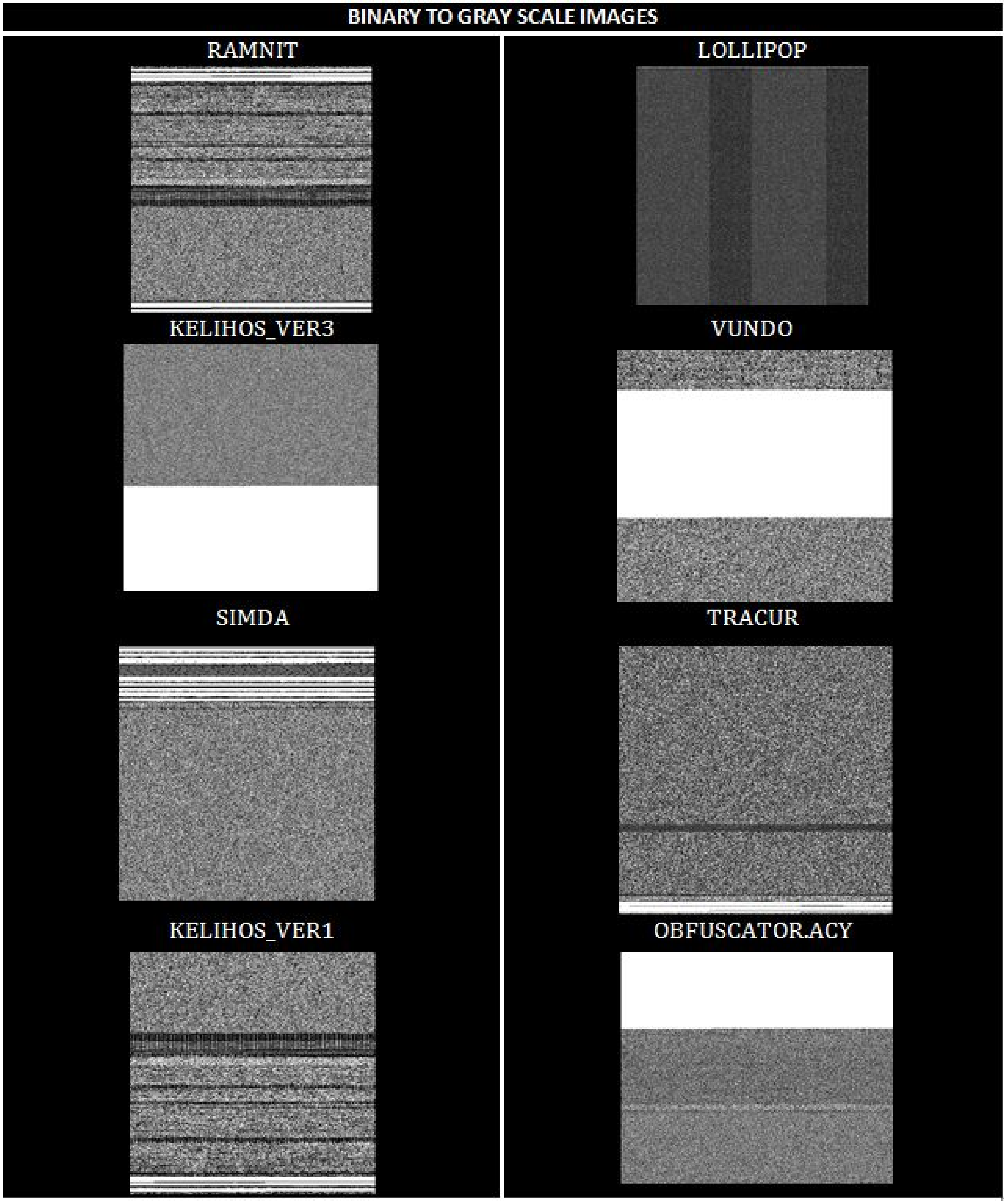
Bytes to Image Conversion: To use our .bytes as input for our deep learning model we convert malware bytes files into image files using "NATRAJ Bytes to Image" algorithm which converts bytes file of specific size into image file of constant size.



```
# Nataraj et al. file size to width table
images_dir = r"D:\Natraj_TestData"
bytes_file = concat_path(bytes_dir, file)
with open(bytes_file, 'r') as f:
    lines = f.read().splitlines()
    first_addr = lines[0].split()[0]
    last_addr = lines[-1].split()[0]

    file_size = int(last_addr, 16) - int(first_addr, 16)
    print('{}: {}kB'.format(file, file_size / 1024))
    if (file_size < 10 * 1024):
        bytes2png(file, 32)
    elif (file_size < 30 * 1024):
        bytes2png(file, 64)
    elif (file_size < 60 * 1024):
        bytes2png(file, 128)
    elif (file_size < 100 * 1024):
        bytes2png(file, 256)
    elif (file_size < 200 * 1024):
        bytes2png(file, 384)
    elif (file_size < 500 * 1024):
        bytes2png(file, 512)
    elif (file_size < 1000 * 1024):
        bytes2png(file, 768)
    else:
        bytes2png(file, 1024)
```

After conversion from bytes to image the different classes of malware in the form of images are as follows:



Resizing of Image files: We then resize the image file into 224*224*3 size to reduce the redundancy from our dataset.

Conversion of Images into Tensors: In order to use image files as input in our deep learning model we convert our malware files into tensors.

3.3 Deep Learning Model

Convolutional Neural Network: A convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. To train our malware classification model we use VGG 16 CNN architecture in our CNN model.

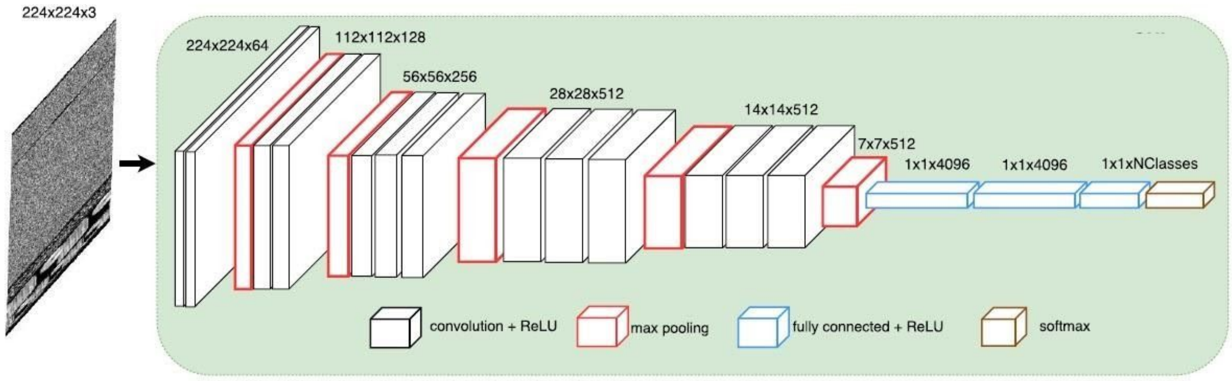
Naive CNN Approach: As an initial step to the malware classification, a naive CNN architecture with 5 layers was implemented. The architecture was inspired by multiple state of the art Cifar-10 architectures, given that the dataset is also a 10-Class problem.

The input to the cov1 layer is of fixed size 200 x 200 Grayscale image. The image is passed through three convolutional layers, where the filters were used with a receptive field: 3×3. The convolution stride is fixed to 1 pixel. Spatial pooling is carried out by three max-pooling layers. Max-pooling is performed over a 2×2 pixel window, with stride 2.

One Fully-Connected (FC) layer has 4096 channels, while the second performs 10-way classification and thus contains 10 channels (one for each class). The final layer is the soft-max layer.

VGG-16 Transfer Learning Approach:

VGG16 Architecture



VGG-16 Architecture Description: VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous models submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU.

The input to the cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time. [10].

The hyperparameter settings are as follows:

Momentum	0.0005
Learning Rate	0.001
Epochs	25
Weight Decay	0.9

Transfer Learning: Transfer learning is an optimization that allows rapid progress or improved performance when modeling the second task. Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned. We use transfer learning concept in our deep learning where we initialize our deep learning parameters with pre trained weights which are trained on random image dataset which help in good generalization and fast convergence.

3.4 Validation Techniques

Holdout Cross-Validation: The holdout method is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set. Usually in holdout cross validation technique the dataset divides into 90% training and 10% for validation. We run 25 epochs using "HOLD OUT CROSS VALIDATION" technique on both the Naive CNN model & Vgg16 model to get the following results.

Model	*Accuracy*	*Loss*
Naive CNN	64.56%	0.2672
VGG16	97.56%	0.0003

K-Fold Cross Validation: The k fold cross validation is a technique in which training data is divided into k number of folds and then we do training on k-1 folds and validate on 1 fold iteratively. We take the value of k=10 run 25 epochs on each fold and get the following result.

Model	*Validation Accuracy*	*Validation Loss*
VGG16	99.01	0.0001

As we can see we got better results by using k-fold cross validation on VGG16 transfer learning approach.

3.4 Testing

Both the naive CNN model, and VGG16 transfer learning model were tested multiple times (each one after tuned settings) on kaggle using log-loss function. The results are as follows:

	Model	*Kaggle Score*
1	Naive CNN	1.17
2	Naive CNN after tuning	1.06
3	VGG-16	0.68
4	VGG-16 after tuning	0.45

3.5 Semi-Supervised Learning

To take advantage of both the labeled and unlabeled dataset, we used a semi-supervised method. For initial experiments we used a natural cluster-then-label method to label the unlabeled data which will be used to further train the final model.

Cluster-then-Label :

1. Input : Labeled and Unlabeled data.
2. Choose a cluster algorithm C, and a supervised learning model L,
3. Cluster each data sample (label and unlabeled mix)
4. For each cluster *i,* train a supervised model L*i* using the initial labeled instances in that cluster.
5. Then finally assign labels to unlabel instances using that learned model *Li*.
6. Output: Unlabeled data with Labels.

4. EXPERIMENTAL SETUP

4.1 Hardware

The hardware used in our final year project is as follows

Categories	**Specifications**
Processor	Core i7 7th Generation
Clock Speed	3.4 GHz
GPU	NVIDIA RTX 2070
Ram	16 GB

4.2 Software

The software used in our final year project is as follows

Categories	**Specifications**
IDE	Jupyter Notebook
Machine Learning Library	Pytorch

4.3 Malware Dataset

About Dataset: We downloaded our dataset from a competition which was posted on Kaggle in 2015 named as "MICROSOFT MALWARE CLASSIFICATION CHALLENGE (BIG 2015)" by Microsoft.

Dataset Description: The dataset contains a set of known malware files representing a mixture of 9 different families. Each malware file has an Id, a 20 character hash value to uniquely identifying the file, and a Class Name, and an integer representing one of 9 family labels to which the malware belongs:

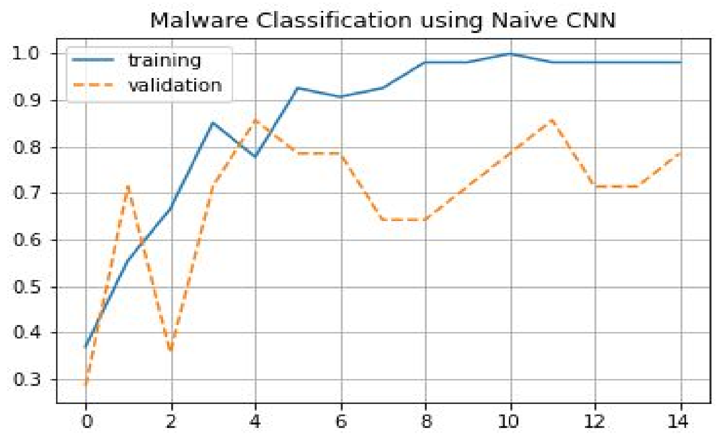
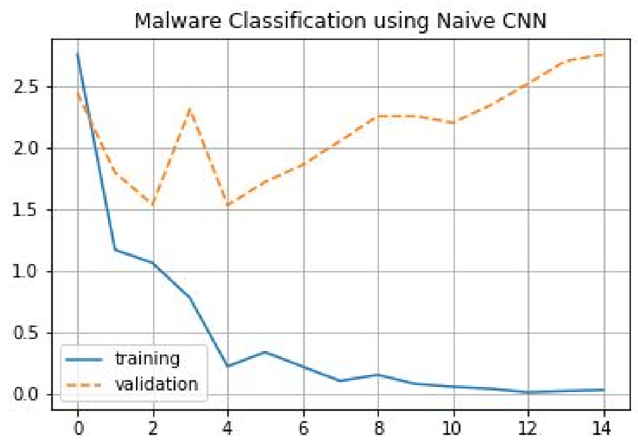
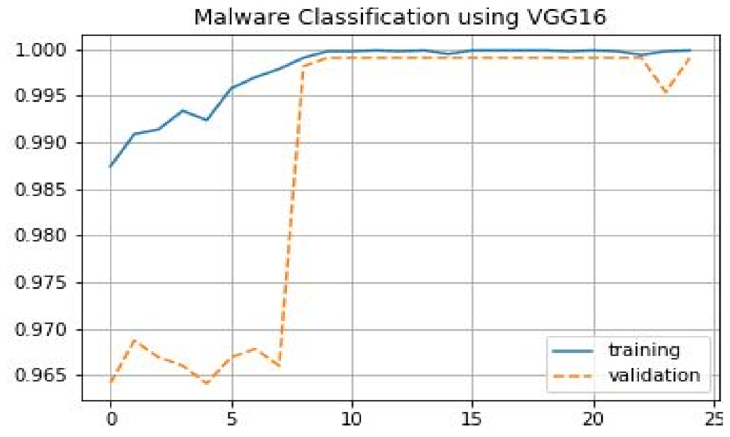
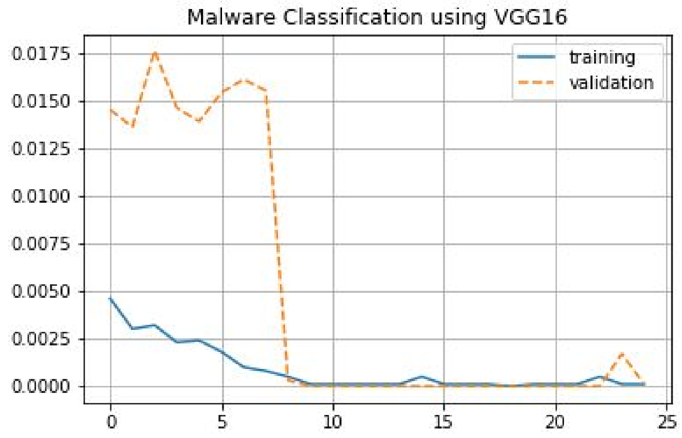
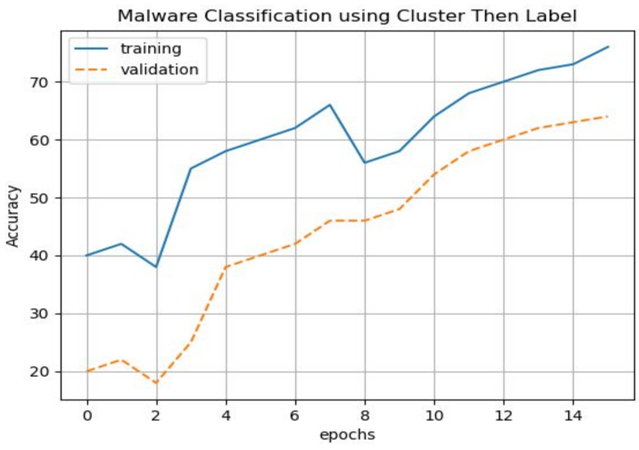
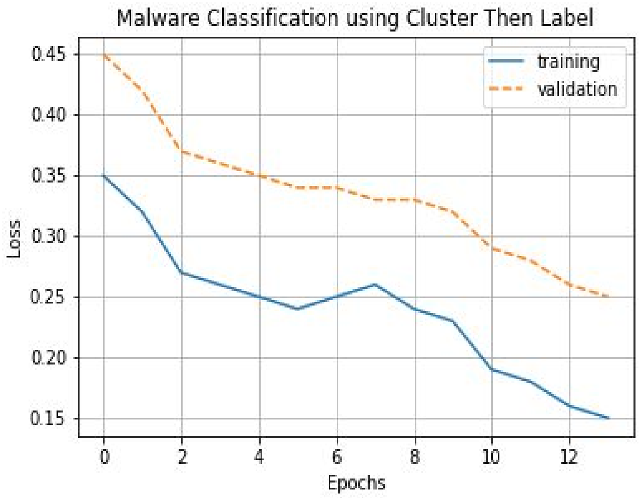
1. Ramnit
2. Lollipop
3. Kelihos_ver3
4. Vundo
5. Simda
6. Tracur
7. Kelihos_ver1
8. Obfuscator.ACY
9. Gatak

The Whole Dataset contain two folders test and train which contain two types of file for each malware .asm and .bytes file

Train Dataset: Train folder contain 10,868 asm files and 10,868 bytes file (total 21,736 files) with a .csv file which contain label of each malware file

Test Dataset: Test folder contain 10,873 asm files and 10,873 bytes file
(total 21,746 files).

5. RESULTS

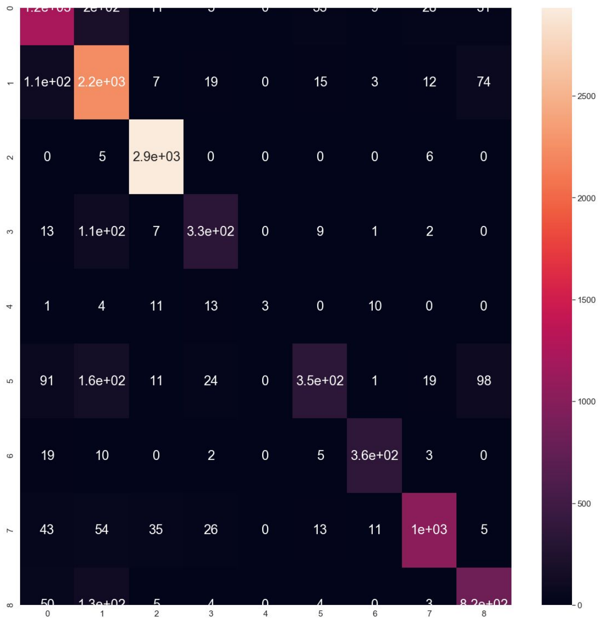
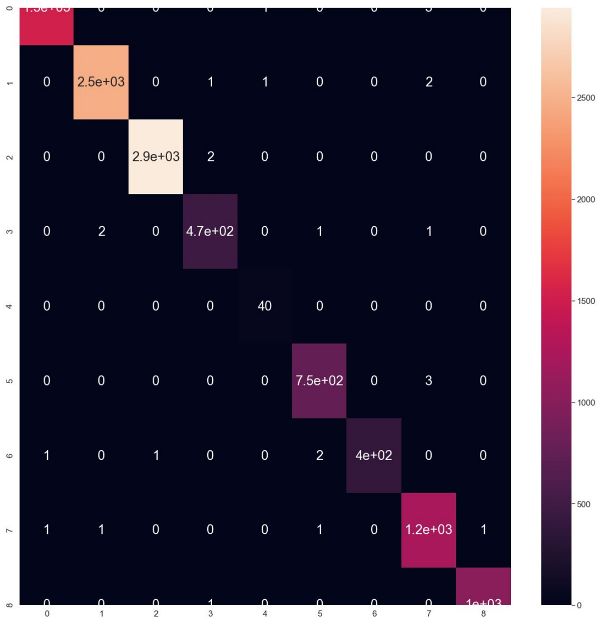
Accuracy	Loss
	
	
	

6. RESULTS DESCRIPTION

Naive CNN: In our basic model, with the learning rate set to 0.001, at first the model suffers to find global minima, but after some epochs the graph becomes stable. Atlast, the training accuracy reached 86% while validation accuracy reached 64%, which was not acceptable as it represents that the model is overfitting the training data.

VGG-16 Model: After research, transfer learning on vgg-16 was implemented which performs quite well on both the training and validation data, giving 99.97% training accuracy & 99.01% validation accuracy.

For both the model, the confusion matrix are given as follows:

Naive CNN	*VGG16*
	

Cluster-Then-Label: As our initial experiment to provide a semi-supervised model, the cluster-then-label approach was chosen, providing 75% training accuracy & 60% validation accuracy. The results of the model suggests that it needs to be tuned precisely for the given problem.

7. CONCLUSION AND FUTURE WORK:

Recent deep learning approaches have been successful in capturing the underlying patterns in images, such as Convolutional Neural Networks (CNN). To capture the underlying patterns, we used VGG 16 (CNN) architecture, a state-of-the-art architecture for image classification, after we converted malware binaries to grayscale images. As labeled data is unavailable or expensive that is why we propose a semi-supervised deep learning approach in our research project through which we will be able to train our model using unlabeled data by using semi-supervised learning algorithms. For that, we used a cluster than label approach to cater to the issue of lack of labeled data.

Furthermore, in the future, we will tune the existing semi-supervised algorithm to get better results, after that we will use Pseudo labeling and one more semi supervised learning approach other than cluster-then-label to get more labeled data. The labeled data which will be produced using these semi-supervised learning approaches will be used to enhance the accuracy of our CNN model because the greater the labeled data the better accuracy of CNN model. Atlast, we will compare the performance of all the semi-supervised learning approaches..

8. REFERENCES

[1] Xiaojin Zhu, Andrew B. Goldberg, Introduction to Semi-Supervised Learning, Morgan & Claypool Publishers, University of Wisconsin, Madison, 2019.

[2] Mehdi Sajjadi, Mehran Javanmardi and Tolga Tasdizen, Regularization with Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning, University of Utah, 14 June 2016.

[3] Gao Yiping, Gao Liang, Li Xinyu, Yan Xuguo, A semi-supervised convolutional neural network-based method for steel surface defect recognition, University of Science and Technology, Wuhan, 430074, China , 2019.

[4] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, Kevin McGuinness, Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning, Dublin City University, 2020

[5] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521, no. 7553 (2015): 436-444.

[6] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." In 2017 International Conference on Engineering and Technology (ICET), pp. 1-6. IEEE, 2017.

[7] Sagi, Omer, and Lior Rokach. "Ensemble learning: A survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8, no. 4 (2018): e1249.

[8] Drew, Jake, Michael Hahsler, and Tyler Moore. "Polymorphic malware detection using sequence classification methods and ensembles." EURASIP Journal on Information Security 2017, no. 1 (2017): 2.

[9] Peikari, M., Salama, S., Nofech-Mozes, S. *et al.* A Cluster-then-label Semi-supervised Learning Approach for Pathology Image Classification. *Sci Rep* 8, 7193 (2018).

[10] VGG16 - Convolutional Network for Classification and Detection (neurohive.io).

[11] Dong-Hyun Lee,The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks,Nangman Computing, 117D Garden ve Tools, Munjeong-dong Songpa-gu, Seoul, Korea

[12] Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A.,Vincent, P., and Bengio, S. Why does unsupervised pre-training help deep learning?. The Journal of Machine Learning Research, 2010, 11: 625-660.