Malware classification using Malimg dataset

Ali El-Sherif [a], Basma Abd-Elwahab [b], Abdulrahman Ahmed [c], Abdelrhman Rezkallah [d]

[a] University of Ottawa, Giza, Egypt, aelsh054@uottawa.ca

[b] University of Ottawa, Gharbia, Egypt, babde014@uottawa.ca

[c] University of Ottawa, Fayoum, Egypt, aahme275@uottawa.ca

[d] University of Ottawa, Alexandria, Egypt, arezk095@uottawa.ca

Abstract

Malware is a software program that is designed to damage your devices and software, so classifying malware images into the known types helps in protecting systems and devices. As the number of malware attacks increase it became much harder to detect malware types manually, so making a model to detect malware image types is better than classifying them manually. The Malimg dataset contains 25 classes, this dataset has been split into training, testing and validation datasets. AlexNet was used as a baseline model, VGG16 model and ResNet-101 models have been used to classify the Malimg dataset, class weights were also applied to handle imbalanced data. In evaluation, macro average scoring for F1-Score has been chosen to evaluate the models, ResNet-101 was the champion with 94% macro average score, and finally has been deployment on streamlit to make an interactive malware image classifier. Classifying malware types, gives the ability to understand the impact of malware attacks and choose the best way to protect systems, devices and sensitive data.

Keywords: Malimg; Image-based Malware classification; deep learning; ResNet-101; Streamlit.

1. Introduction

Malwares can be harmful to any device, and they need to be classified to the known types after detection to prevent system breaches and capturing of sensitive data. Building a system from scratch can be a hassle for such classification systems. Image-based malwares should be classified automatically instead of manual classifications by experts which can vary in accuracy due to different levels of knowledge and intellectual skills between them.

2. Literature review

2.1. MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things (Sudhakar & Kumar, 2021).

2.1.1 Summary:

Security experts are endeavoring to make a method that precisely perceives all malware. In this article, the author has proposed a one-of-a-kind convolution mind network-based malware-gathering method. Without featuring the planning, sorting out, or even the advanced dubious strategies used to make the disease, the MCFT-CNN model sees the dark malware test.

Considering its qualities and capacities, the malware might be arranged into many kinds, like worms, diversions, spyware, ransomware, infections, secondary passages, and adware. A new malware assortment presents extra snags for the online protection specialist to distinguish precisely.

Text and picture information is altogether different from the information gathered and made by malware tests. Nataraj et al put out an original technique for malware ID and grouping using visual qualities. They changed the parallel executable's construction into two-layered, greyscale pictures. Afterwards, the AI calculation for malware discovery was prepared to utilize these qualities. Without including designing or past information on twofold code investigation or picking apart, the MCFT-CNN model might foresee unidentified malware types.

By using polymorphic change and morphological jumbling, it can unequivocally identify malware variations intended to get past antivirus programs. Also, the model is more exact and quicker in foreseeing known malware.

A MCFT-CNN model was recommended in this review to sort malware tests into malware families. With the Adam enhancing compiler, it has a 99.05% exactness rate, while prepared utilizing the traditional learning strategy has a 99.22% precision rate. The model likewise accomplished a critical accomplishment with its 5.14 ms gauge time for obscure malware tests.

2.1.2 Research goal:

This paper has introduced an MCFT-CNN model for classifying malware tests into malware families. The advantage of utilizing a profound learning-based CNN model is that no component designing is required. Besides precision, proposing a model that characterizes new malware low time. Their methodology is to outperform past state of art concentrates on malware picture characterization involving profound learning as far as exactness and expectation time.

2.1.3 Methodology:

Deep learning is a subfield of AI that falls under the umbrella of computerized reasoning. In profound learning, input is taken care of into the model, which then, at that point, registers through layers to anticipate or arrange obscure information. The layers of the profound learning model depend altogether on counterfeit brain frameworks for information preparation. A transfer learning procedure was utilized to order malware parallels. The malware doubles are displayed in greyscale. Pictures of malware family varieties show underlying likenesses. Accordingly, the order might be concluded that the infection pictures are utilized as picture classification. For this situation, the entire data of malware tests are used.

2.1.4 Experiments and results:

The model is prepared to utilize a standard learning technique utilizing benchmark datasets (MalImg and Microsoft datasets). A few examples are moreover muddled and incorporate a powerful motor inside the code. MCFT-CNN outflanks ResNet50 with regards to ordering varieties of a certain malware family, like CLOP, Swizzor.gen! E, as well as Wintrim.BX.

(Preciseness:.96 to 1) The classification report portrays Examination I and Trial III, each utilizing Adam and NAdam optimizers. Our model anticipated obscure examples in 5.14 ms, which is altogether quicker than the present status of-the-craftsmanship research on the Huge 2015 dataset.

2.1.5 Strengthen and weakness:

The model's strengths include 99.05% accuracy with the Adam optimizer and 99.22% accurateness with the NAdam optimizer when constructed using the traditional learning technique. The model also achieved a crucial success in the anticipation time of cryptic malware testing, which is 5.14 ms. Furthermore, the model

obtained 99.18% accuracy with the Adam optimizer and 99.10% preciseness with the NAdam optimizer in the interchange learning technique.

Regardless, the limitation is, that there is a limit of employing the model benefits from uniform image size. The author should have used the spatial pyramid pooling layer, which may handle any size information image, however he has stated that he will consider this in the future.

2.2. Malware Classification Using Automated Transmutation and CNN (Agarwal et al., 2021).

2.2.1 Summary:

Malware is a piece of software that damage or cause harmful for the computers, this software mainly used to steel sensitive information.

Due to the increases of malicious softwares we need to analyze them to develop a software that can detect the malicious attack before it happens, also it`s essential that detect the malware type, so now days the machine learning and deep learning can be used to detect the pattern of malware and block any malicious attack.

In this conference paper the authors proposed a solution to study the pattern of the malware it`s a binary data and show it as image to classify malware images using a deep learning CNN architecture pretrained model to extract the different malware signatures

The authors here use a pretrained model to build his solution VGG16 and use the weight of this model to build his approach.

2.2.2 Research Goal:

This conference paper aims to classify the malware types using malware images using deep neural networks pretrained model VGG16, set the suitable parameters for model training, remove some layers and evaluate the model performance, the authors use accuracy as evaluation matrix.

The authors started their work by a literature review for the past related works for classical supervised machine learning techniques such as K nearest neighbor and support vector machine model (SVM) which achieve on 14 types of malwares with average accuracy 88%.

And compare the previous result to his approach that achieve accuracy 98.97%.

2.2.3 Methodology:

The authors proposed a methodology that meets the research goal which is to find a suitable method to determine the malware type starting with visualization as known the malware is present in executable format so looking at binary file and divided it for each 8 bits that help to visualize the malware type as image and detect the malware signature here the authors use gray scale image the size is 224*224,

The proposed model uses a deep neural network pretrained model VGG16 and use transfer learning to apply his approach the loss function used is categorical cross entropy this loss function gives probability that can distinguish between two classes of each other, the activation function used in the convolutional layer is ReLU and the activation function of the output layer is softmax to give the probability for each class.

2.2.4 Experiments and results:

In this paper, the authors provided an applied result, for the experiments they used a malware images dataset which contains 9339 images from different 25 malware families, split the data into 90% training (8405) images and 10% testing (934).

For experimental purpose, the authors used Google Colab with GPU enabled. Configuration of the virtual machine was as follows: Dual Core (TM) Intel Xenon CPU (2.30 GHz) with 12 GB RAM and Nvidia P40 GPU 12 GB RAM. Operating System used was Ubuntu 20.04 64bit they trained the model on number of epochs = 20 and the Batch size= 10000

The authors apply many different models such as GIST with SVM and achieve 93.23% also M-CNN and achieve 98.52% and the VGG16 his chosen model achieve 98.97%.

2.2.5 Strengthen and weakness:

This paper achieved high accuracy in classifying the malware types, one of the strengthen points is that the authors used a very simple approach transfer learning to achieve his result and didn`t use any complex feature engineering to classify the malware images but they didn`t mention why they use transfer learning also, they used accuracy as evaluation matrix and they didn`t mention that the data is imbalance or not because if it`s not imbalance the accuracy metrics didn`t fit well.

2.3. IMCLNet A lightweight deep neural network for Image-based Malware Classification (Zou et al., 2022).

2.3.1 Summary:

Malware is a software program or code that can be harmful for your computing device, it specifically designed to damage, distribute, or gain an unauthorized access to your device. By the time, the number of malwares increased, so, detecting malware type is a very important challenge. To solve this challenge, it's essential to detect the malware type by the machine learning techniques.

The authors proposed a machine learning model to classify the malware images, which is the lightweight malware classification model (IMCLNET) that doesn't need complex feature engineering, large domain knowledge, pre-training parameters and data enhancement. By studying the impact of the images size, the authors said that the images size can affect the performance of the classification model by decreasing or increasing the classification model performance, but the accuracy doesn't change as the performance. The prediction time of the IMCLNET model also affected by the images size. So, it's a great challenge to reduce the prediction time and the impact of images size without affecting the classification model accuracy.

2.3.2 Research goal:

This paper aims to detect the malware types using malware images using the lightweight malware classification model (IMLCNET) using deep neural networks, set the suitable parameters for model training, evaluate the model performance, and increase the classification model accuracy without affecting the model performance and predictions.

The authors started their work by a literature review for the past related works for classical supervised machine learning techniques such as support vector machine model (SVM) which doesn't achieve a high performance for classifying the malware types, image-based methods, which are about dealing with the malware images on gray scale to classify them with the DenseNet model, this method are not sufficient and the detecting efficiency is very low.

The lightweight malware classification model has an efficiency on detecting the malware type with high model performance, limited preprocessing and feature engineering and without data augmentation.

2.3.3 Methodology:

The authors proposed a methodology that meets the research goal which is to find a suitable method that can classify the malware images. They performed malware images visualization as a grayscale image because it doesn't require complex feature engineering, complex feature extraction and expert domain knowledge. After applying the grayscale to the images, resize the images to a uniform size which is 224 * 224. While normalizing the data, most of malware images may lose the most important features, but most of them save their texture features and layouts. The similarity of layouts between the two images indicates that they are from the same family, then we can use these images in malware classification task. The IMCLNET which consists of full convolutional layer, coordinate attention layer, Depth wise separable convolutional layer, and pooling layer can classify malware images and calculate the number of parameters and weight accuracy. The attention mechanism helps in malware classification by capturing the positional information and finds the relationships between the channels to generate the attention maps. The Depth wise separable convolutional layer applies only one filter for each input channel, and makes the model performance efficient by reducing the complexity of the classification model. Finally, the global context embedding designed to work at different stages to capture different feature embeddings of malware images while extracting features from the malware images.

2.3.4 Experiments and results:

In this paper, the authors provided an applied result, for the experiments they used a malware images dataset which contains 9339 images from different 25 malware families, and a dataset contains bytecode files from Kaggle which belongs to a competition published in 2015. The bytecode files used to generate the malware images for classification task.

The authors used 75% of the data for training and 25% for testing, using python 3.9 environment with pytorch and Nvidia 12GP GPU. They used 50 epochs and learning rate = 0.0038 for training the model. They used FLOPs metric to evaluate the model and calculate the model complexity. Then they used accuracy score, precision, recall and F1-score for model evaluation.

They implemented several models to compare with IMCLNET model such as SVM, Adaboost, and several classification techniques, and the IMCLNET model with cross validation was the best one with 99.272% accuracy score for the image dataset, and 98.666 % accuracy score for the BIG2015 dataset.

2.3.5 Strengthen and weakness:

This paper achieved high accuracy in classifying the malware types, one of the strengthen points is that the authors used cross validation, filtering mechanism and classifying the images without performing complex feature engineering on the images. But they didn't mention the disadvantages of IMCLNET model and the effect of hyperparameter tuning of the model.

2.3.6 Solution:

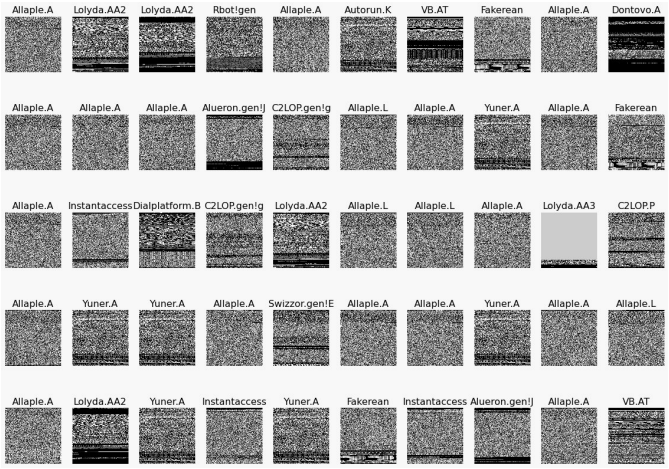
By using Malimg dataset which contains 9339 malware images that belong to 25 families/classes, a pre-trained model will be used with our own architecture, probably VGG-19 which is a 19 layers depth convolutional neural network that can classify images into 1000 object categories. The data should be deployed and trained with a fine-tuning to be done. Cross-validation will also be used and the data will be split into 80% train, 10% validation and 10% test. The classification will be evaluated using common metrics such as accuracy, recall, precision and F1-score. There will be a deployment and data representation to be made on streamlit.

3. Research objectives/questions

Malware is any malicious code or a program that can be harmful to the computer. There are many types of malwares, and it's essential to detect these types to prevent their breaches to keep the data and the system private and secured. The aim of this project is to classify the detected malware image used from the Malimg dataset through an interactive application on streamlit.

4. Dataset

The Malimg dataset consists of 9339 images and 25 classes. The dataset contains the family/class of the malware and the malware type. The data has been split into training, testing and validation sets. Figure (1) shows random samples of malwares from different classes while figure (2) shows the dataset's feature percentage distribution. In table (1), each family/class is shown with its corresponding malware type.



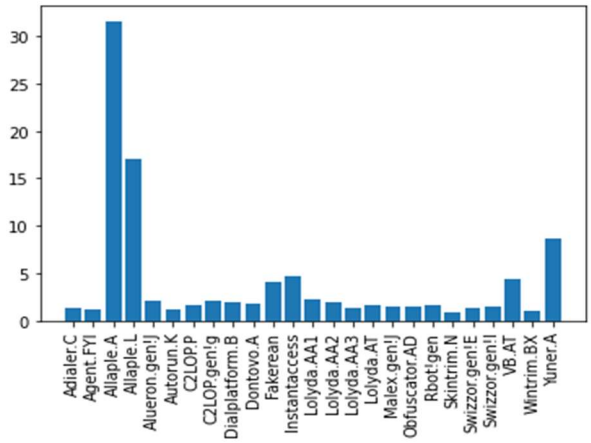


Fig 1. random samples of malwares from different classes.

Fig 2. dataset's feature percentage distribution.

Family/ class	Type	Family/ class	Type	Family/ class	Type
Adialer.C	Dialer	Dontovo.A	Trojan Downloader	Obfuscator.AD	Trojan Downloader
Agent.FYI	Backdoor	Fakerean	Rogue	Rbot!gen	Backdoor
Allaple.A	worm	Instantaccess	Dialer	Skinterm.N	Trojan
Allaple.L	Worm	Lolyda.AA1	PWS	Swizzor.gen!E	Trojan Downloader
Alueron.gen!J	worm	Lolyda.AA2	PWS	Swizzor.gen!I	Trojan Downloader
Autorun.K	Worm.AutoIT	Lolyda.AA3	PWS	VB.AT	worm
C2LOP.P	Trojan	Lolyda.AT	PWS	Wintrim.BX	Trojan Downloader
C2LOP.gen!g	Trojan	Malex.gen!J	Trojan	Yuner.A	worm
Dialplatform.B	Dialer				

Table 1. family/class and type of each malware.

5. Methodology

The experimentation was done mainly using TensorFlow Keras library on Google Colab (cloud development environment). For the experiment, the benchmark was based on images' dataset of size 1.2 GB that was trained on AlexNet, VGG16 and Resnet101, and all these models were trained in 50 epochs and the data was split into 80% Training, 10% Validation and 10% Testing ratio as showed in the system's architecture in figure (3). Further, transfer learning technique was applied using the pretrained models. Fine tuning was done to improve the accuracy. Early checkpoint was created that monitored the change in the validation loss and patience was set to 2 iterations to find the

best model. To create a better model, the learning rate was reduced by the factor of 0.01 if validation loss does not change with epochs. For the compiling model, Adam optimizer was used to set the best weight and learning rate.

The concept of transfer learning is to use the pretrained model of one task as a foundation for other tasks. The main idea was to take the best pre-trained models and use it in a specific situation and use them in another setting by tuning the parameters that would be the best for that situation. ResNet, Oxford VGG and Google Inception are the commonly used pretrained models which were the winners of the ImageNet competition.

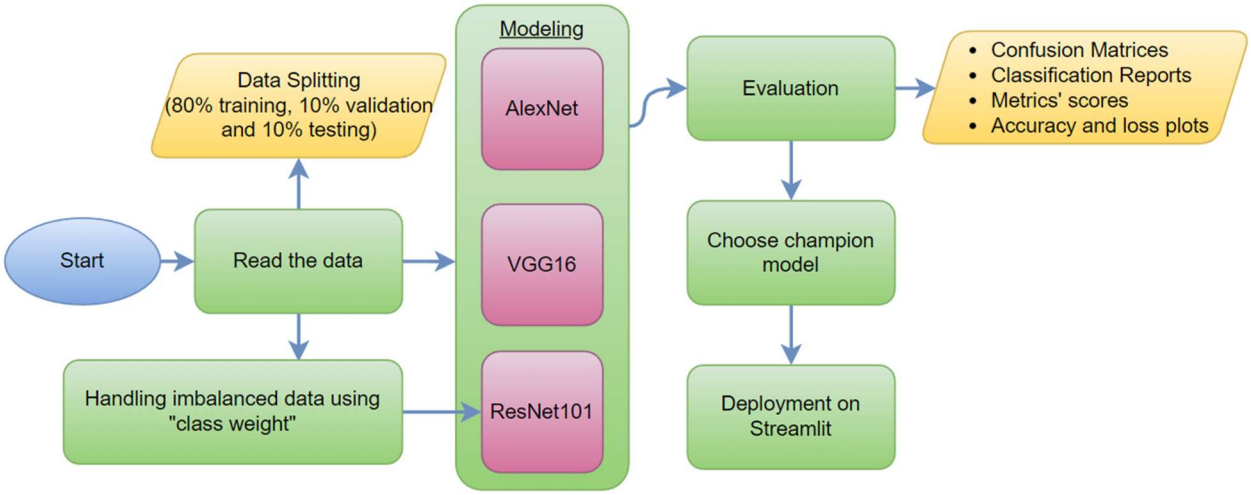


Fig 3. System architecture.

Alexnet has won the Imagenet large-scale visual recognition challenge in 2012 using its eight layers with learnable parameters. The model consists of five layers with a combination of max pooling followed by 3 fully connected layers and ReLU activation function in each of these layers except the output layer as shown in figure (4) (Iandola et al., 2016).

Generally, it was found out that using ReLU activation function accelerated the speed of the training process by almost six times. Nevertheless, dropout layers have been used, that prevented their model from overfitting. Further, the model is trained on the Imagenet dataset. The Imagenet dataset has almost 14 million images across a thousand classes (Recht et al., 2019). In this project, 2x2 convolutional layers with ReLU activation function, max pooling 2D with pooling size 3x3 and a softmax activation function to the output layer were added to the original model.

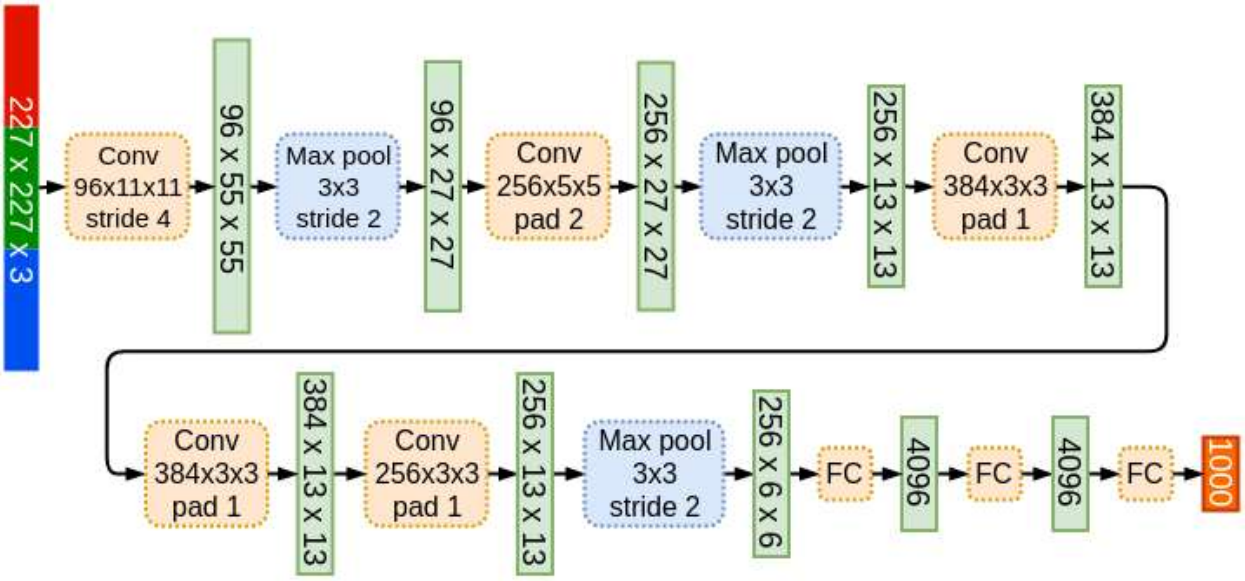


Fig 4. AlexNet model structure (Agarwal, 2022).

VGG-16 is a CNN based deep learning architecture which won ILSVR (ImageNet) competition in 2014. It has 16 deep layers in total consisting of 5 convolution layers, 5 max pooling layers, 3 fully connected layer, and a dense layer as showed in figure (5). For training the proposed model, an image of size 224×224 was passed into the different layers of VGG16, adding to that, global average pooling layer and 2 dense layers were added to the base model for classifying malwares into 25 different categories.

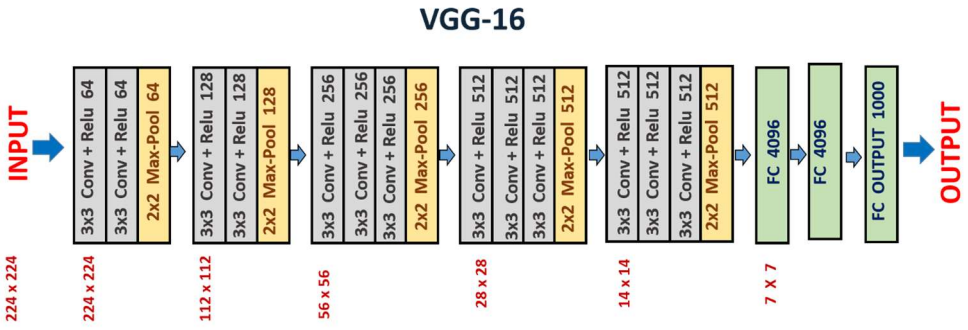


Fig 5. VGG16 model structure (Khandelwal, 2020).

ResNet is a deep convolution neural network that won the ILSVR (ImageNet) competition in 2015. ResNet solved the problem of vanishing or exploding gradient. The ResNet-101 -figure (6)- has 101 deep layers that takes an image of

input size 224×224. For training with ResNet, the standard ResNet-101 model was used by setting it in such a manner that it could classify 25 different malwares. The following were added to the ResNet-101 model which are, max pooling 2D of size 2x2, dropout of 0.5, using ReLU activation functions and softmax activation function for the output.

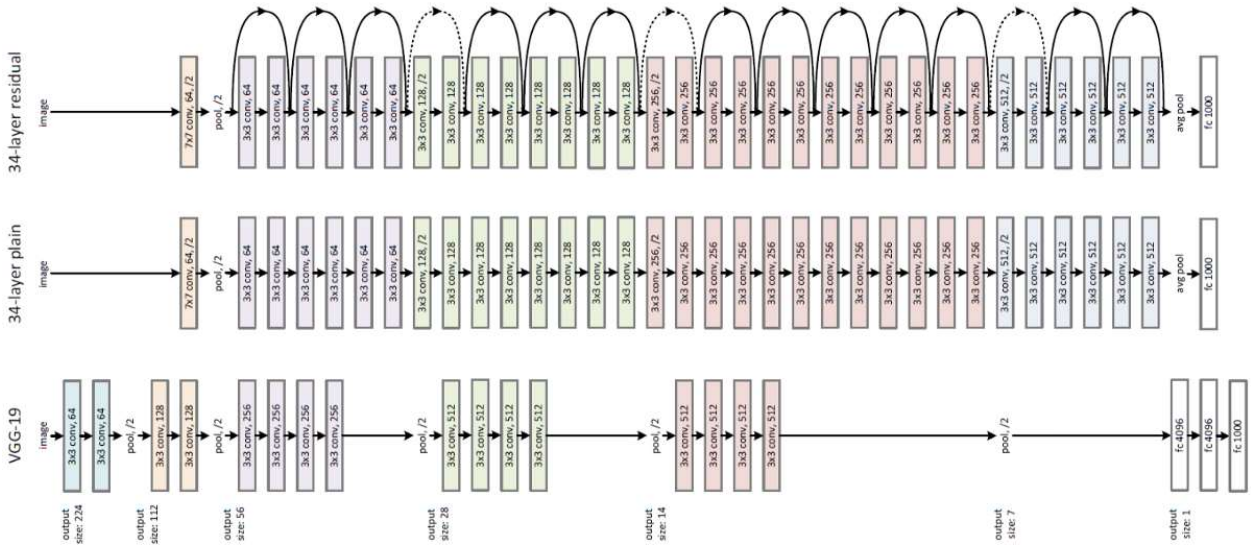


Fig 6. ResNet101 model structure (Tsang, 2019).

Alexnet was applied as baseline model without handling imbalanced data and macro average F1-score was used as the evaluation matrix and after training for 50 epochs and using Adam as optimizer, it has reached 86% in testing as base line model.

To improve this result, VGG16 was applied with the same number of epochs and optimizer and has reached 89% macro average F1-Score and it was noticed that there was a problem of vanishing gradient, to solve this problem, ResNet was used because its architecture has skip connections to overcome vanishing gradient. This model can handle imbalanced data using class weight and apply early stopping to monitor the validation loss with the same parameters 50 epochs and Adam optimizer and it has reached 94% macro average F1 score which is the champion Model.

6. Evaluation

A macro average F1-score has been used for evaluating all models because of the imbalanced of the data.

Hyperparameter tuning was done through:

- Trying different loss functions.
- Trying various activation functions.
- Trying different layers with different number of neurons.
- Trying different optimizers.
- Trying different batch sizes and different epochs.

For Freezing /without freezing, some options have been tried, such as:

- Adding more layers while freezing the architecture layers.
- Adding our own layers while unfreezing the architecture layers.
- Unfreezing some of the architecture layers, freeze some layers while adding others.

6.1 AlexNet model

6.1.1 AlexNet (baseline model)

AlexNet has been set as a baseline model. SoftMax activation function has been added to the output layer. The model was then trained across 50 epochs, with Adam optimizer, and sparse categorical cross-entropy as the loss function and the data for the training is relatively small for the model to learn, with the loss and accuracy values being printed for each epoch.

6.2 VGG16 model

6.2.1 First tuned VGG16 model

Firstly, three layers have been added with 2048, 1024, and 512 neurons each after freezing the VGG16 layers. ReLU was proved to be the most effective among the activation functions that has been tried including leaky

ReLU, and Tanh. Also, SGD optimizer has been used, but it somehow performed poorly, so Adam optimizer is the best, and sparse categorical cross entropy as the loss function with a batch size of 64 and 30 epochs.

6.2.2 Second tuned VGG16 model

In the previous experiment the model's performance was not good enough regarding the overfitting problem, so, the same architecture of the previous experiment has been used. The model has been modified by adding some batch normalization and dropout layers to prevent the model from overfitting.

6.2.3 Third tuned VGG16 model

The model has been trained for 50 epochs, with the loss and accuracy values being printed for each epoch.

6.3 ResNet-101

In this particular approach, the model has seemed to perform well, with the training loss and accuracy both improving over time, and the validation loss and accuracy remaining relatively stable. However, there are a few points where the validation accuracy dips slightly, but overall, the model performance has been stable and was chosen as the champion model.

7. Results

7.1 AlexNet

Nevertheless, the model has achieved a high accuracy on the validation set, with values above 80% for most epochs. The loss values for the training and validation sets are similar, indicating that the model is not overfitting to the training data. The model's performance on testing data was 86% of the macro average F1-score showed in figure (7), which is a relatively lower accuracy compared to other used models. Figure (8) shows the loss curves and accuracy curves while figure (9) shows the confusion matrix of the model.

testing report :

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	1.00	1.00	13
2	0.86	0.95	0.90	296
3	0.89	0.74	0.81	160
4	0.95	1.00	0.98	21
5	0.00	0.00	0.00	12
6	0.92	0.69	0.79	16
7	0.67	0.80	0.73	20
8	1.00	1.00	1.00	19
9	1.00	1.00	1.00	17
10	1.00	0.97	0.99	39
11	1.00	1.00	1.00	44
12	1.00	0.95	0.98	22
13	1.00	1.00	1.00	19
14	1.00	1.00	1.00	13
15	0.94	0.88	0.91	17
16	0.93	0.87	0.90	15
17	1.00	1.00	1.00	15
18	0.93	0.76	0.84	17
19	1.00	1.00	1.00	8
20	0.45	0.36	0.40	14
21	0.58	0.79	0.67	14
22	1.00	1.00	1.00	42
23	0.67	0.73	0.70	11
24	0.87	1.00	0.93	80
accuracy			0.89	957
macro avg	0.87	0.86	0.86	957
weighted avg	0.89	0.89	0.89	957

Fig 7. Classification report of AlexNet.

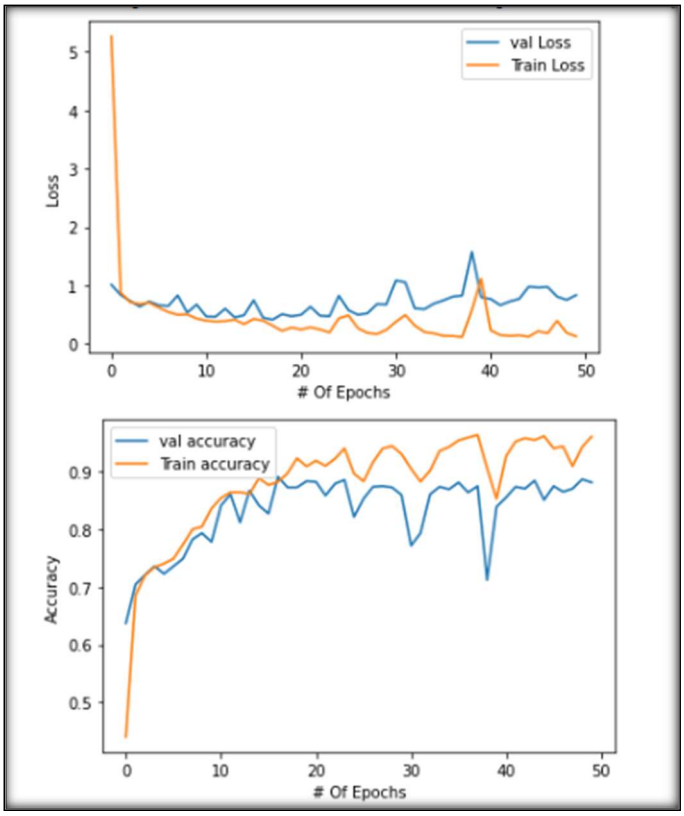


Fig 8. Accuracy and loss curves of AlexNet.

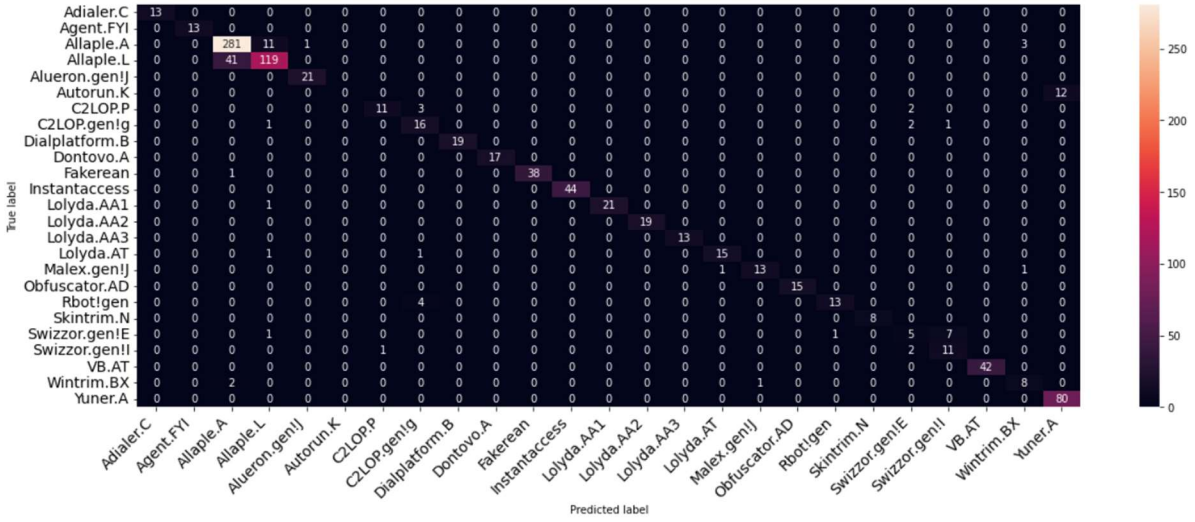


Fig 9. Confusion matrix of AlexNet.

7.2 VGG16 model

7.2.1 First tuned VGG16 model

After the tenth epoch, the training loss keeps getting worse as the number of epochs goes up, while the training accuracy goes up until it reaches its highest value, which is 100%, and the training loss keeps getting smaller. This case is known as an overfitting problem. A macro average F1-score accuracy of the model on the test dataset is 78% and the model misclassified some samples from the class twenty, twenty-one and class five.

7.2.2 Second tuned VGG16 model

The model learned in a better way, regardless there is a high variation in the validation accuracy, the validation loss is not much larger than the training loss, the gap between training and validation accuracy was also relatively small, this new architecture successfully eliminates the overfitting problem and test macro average F1-score is 78% and there is high oscillation.

7.2.3 Third tuned VGG16 model

It also shows that the model is achieving a high accuracy on the validation set, with values above 90% for most epochs. However, the loss values for the validation set are sometimes higher than the loss values for the training set, indicating that maybe the model is overfitting using the training dataset. The model's performance on testing dataset is 89% of macro average F1-score showed in figure (11).

testing report :

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	1.00	1.00	13
2	1.00	1.00	1.00	296
3	1.00	1.00	1.00	160
4	1.00	0.90	0.95	21
5	0.00	0.00	0.00	12
6	0.70	1.00	0.82	16
7	0.81	0.65	0.72	20
8	1.00	1.00	1.00	19
9	1.00	1.00	1.00	17
10	1.00	0.97	0.99	39
11	0.98	1.00	0.99	44
12	1.00	1.00	1.00	22
13	1.00	1.00	1.00	19
14	1.00	1.00	1.00	13
15	0.94	0.94	0.94	17
16	0.88	1.00	0.94	15
17	1.00	1.00	1.00	15
18	1.00	0.94	0.97	17
19	0.89	1.00	0.94	8
20	0.71	0.36	0.48	14
21	0.56	0.71	0.63	14
22	1.00	0.98	0.99	42
23	0.92	1.00	0.96	11
24	0.87	1.00	0.93	80
accuracy			0.96	957
macro avg	0.89	0.90	0.89	957
weighted avg	0.95	0.96	0.95	957

Fig 11. Classification report of 3rd VGG16 model.

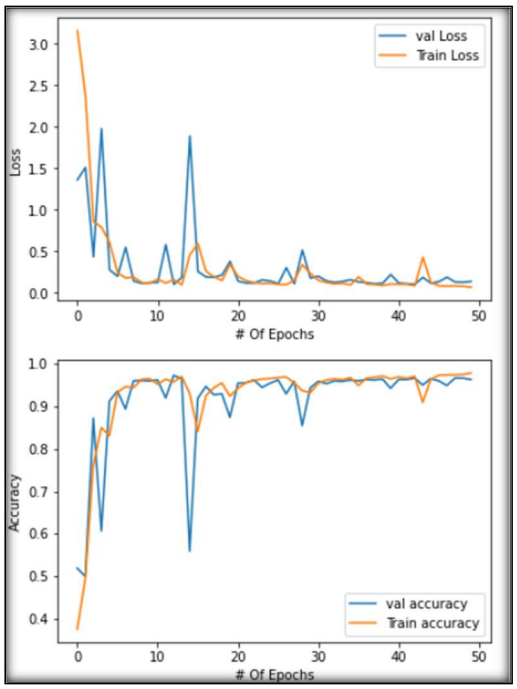


Fig 10. Accuracy and loss curves of 3rd VGG16 model.

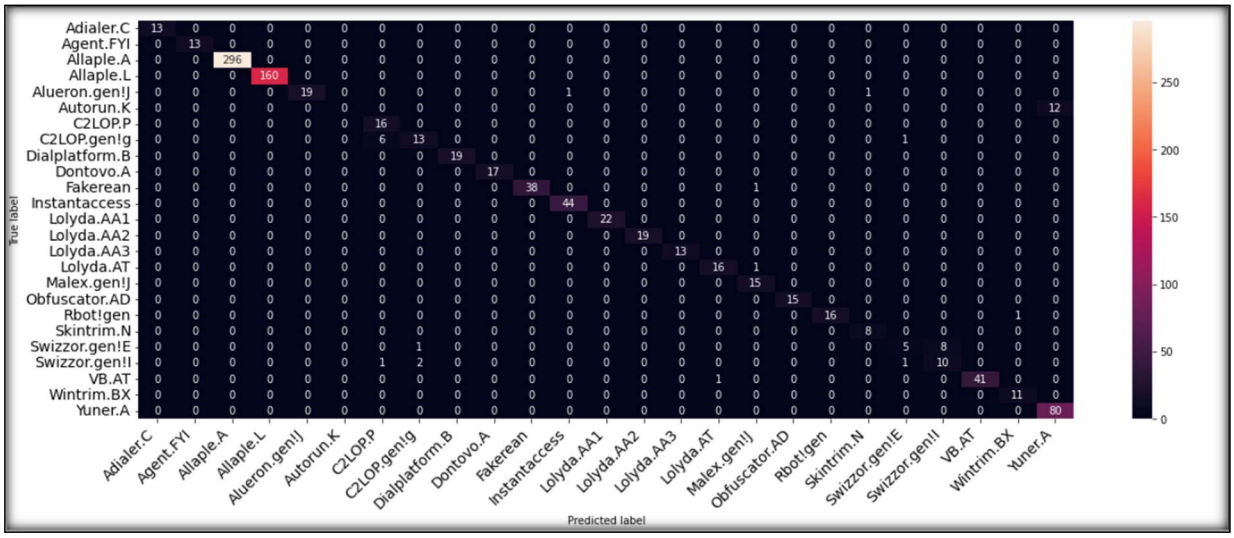


Fig 12. Confusion matrix of 3rd VGG16 model.

7.3 ResNet-101

It has given the highest macro F1-score accuracy on test data 94% as shown in figure (14). The class weight has been used to handle the imbalance which was included in our classes after that early stopping technique has been used to monitor the validation loss across number of epochs. ResNet-101 was the best model because it contains skip connections to combine features from earlier layers to obtain higher level features.

testing report :

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	1.00	1.00	13
2	0.98	1.00	0.99	296
3	1.00	0.98	0.99	160
4	1.00	1.00	1.00	21
5	1.00	1.00	1.00	12
6	1.00	0.94	0.97	16
7	1.00	0.95	0.97	20
8	1.00	1.00	1.00	19
9	1.00	1.00	1.00	17
10	1.00	1.00	1.00	39
11	1.00	1.00	1.00	44
12	0.96	1.00	0.98	22
13	1.00	1.00	1.00	19
14	1.00	1.00	1.00	13
15	1.00	0.94	0.97	17
16	1.00	0.87	0.93	15
17	1.00	1.00	1.00	15
18	1.00	1.00	1.00	17
19	1.00	1.00	1.00	8
20	0.00	0.00	0.00	14
21	0.48	1.00	0.65	14
22	1.00	1.00	1.00	42
23	1.00	1.00	1.00	11
24	1.00	1.00	1.00	80
accuracy			0.98	957
macro avg	0.94	0.95	0.94	957
weighted avg	0.97	0.98	0.97	957

Fig 13. Classification report of ResNet-101.

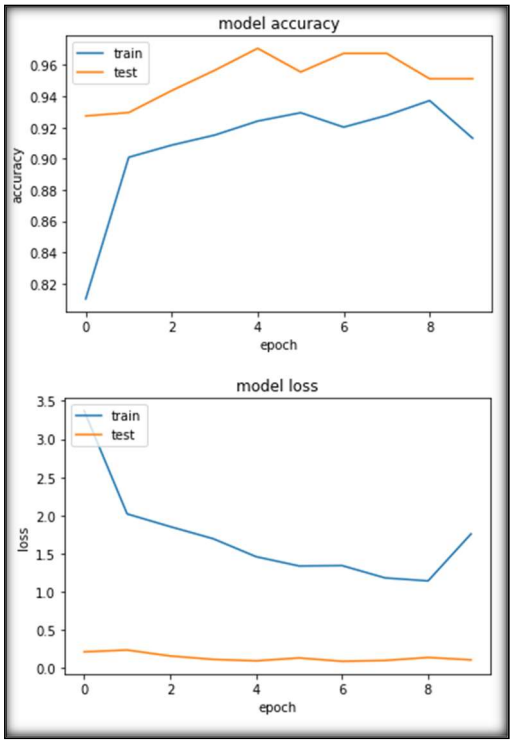


Fig 14. Accuracy and loss curves of ResNet-101.

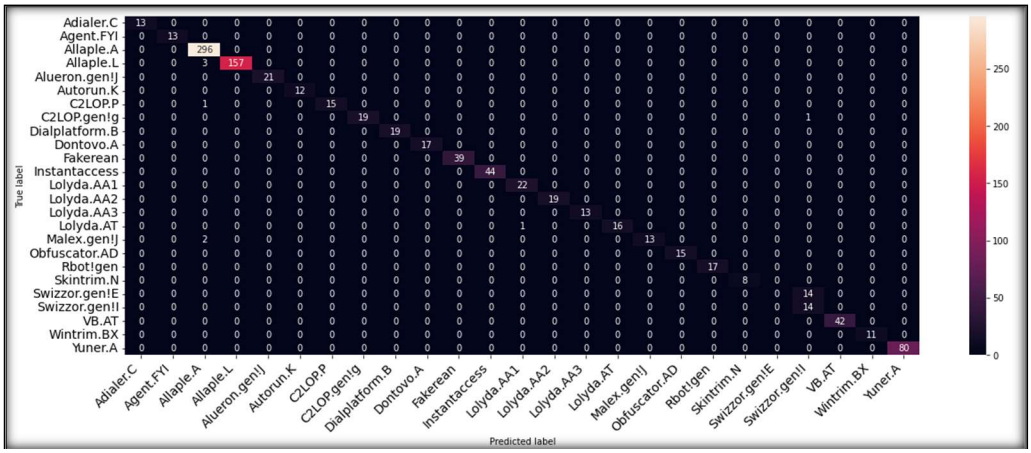


Fig 15. Confusion matrix of ResNet-101.

8. Critical discussion

In AlexNet model, the tuned VGG16 model and the ResNet-101 model, all of the three models misclassified class twenty, class twenty-one and five which are of family Swizzor.gen!I, Autorun.K, and Swizzor.gen!E in figures (16, 17 and 18) respectively.

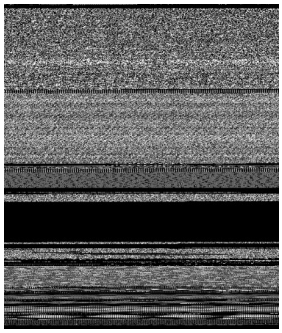


Fig 16. Swizzor.gen!I sample.

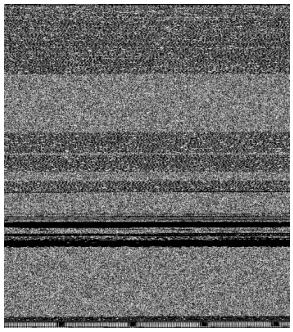


Fig 17. Autorun.K sample.

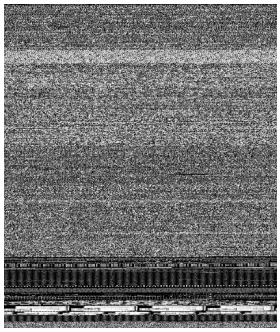


Fig 18. Swizzor.gen!E sample.

Which classified them as Adialer.C, Agent.FYI, Lolyda.AA3 and Dontovo.A families figures (19-22) respectively.

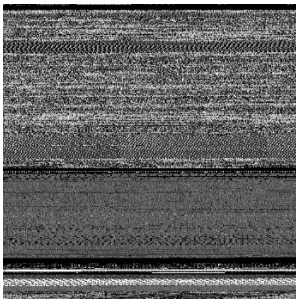


Fig 19. Adialer.C sample.

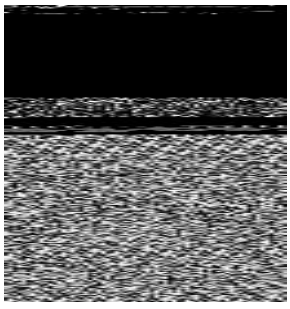


Fig 20. Agent.FYI sample.

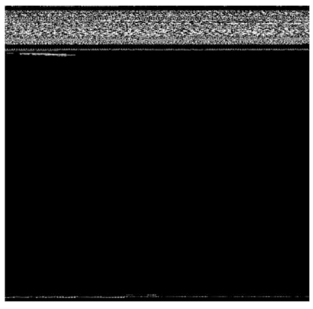


Fig 21. Lolyda.AA3 sample.

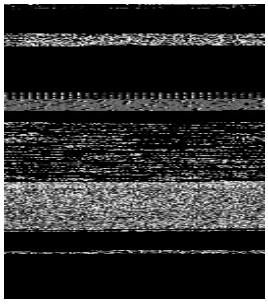


Fig 22. Dontovo.A sample.

9. Deployment

For the deployment part, streamlit framework was used to deploy the champion model which is ResNet 101 by saving the model in h5 format to save all the training weights, load the model, and create a suitable design using HTML and CSS to design the style of the application. The interactive page was designed as a simple easy-to-use application all what it needs is to load the desired image to be classified as in figure (24), and the application would apply all the preprocessing and return the malware name with the classification status (i.e., classified) which is shown in figure (23).

Fig 23. Classification image upload.

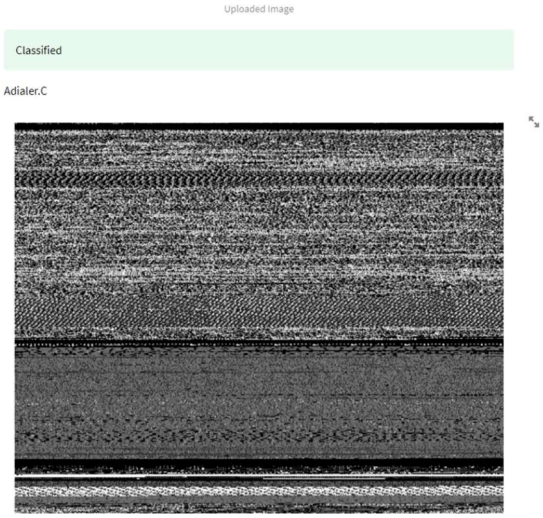


Fig 24. Classification result.

10. Conclusion

Malware images can be really effective in their attacks on many devices that is not suitable to detect them manually as its time consuming, exhaustive and depends on personal skills of the assigned engineer. The Malimg dataset contains 25 classes of different malware families. Three models have been applied to classify Malimg dataset with Adam optimizer and categorical cross-entropy as a loss function, these models are, AlexNet as the baseline model with 86% macro average f1 score, tuned VGG16 model achieved 89% macro average f1 score, and ResNet-101 model has achieved 94% macro average f1 score. The Macro Average F1-Score has been set for evaluation as the data is imbalanced. The champion model is the ResNet-101 model that has achieved high results in classifying Malimg dataset, and has been chosen for deployment using Streamlit to make an interactive user interface to classify the malware images to their right families.

For the future work, a central database should be built to serve multiple users at the same time, also ResNet-101 model will be uploaded to a cloud server using a GPU for more reliability and availability, finally building a dynamic model system to train periodically when the model's accuracy drops because of any changes.

11. Future work

- Build a central database that makes the application able to serve multiple users.
- ResNet-101 model should be built on a cloud server with GPU support to be a more reliable and scalable machine system and be up 24 hours to handle requests more efficiently and accelerate the inference time.
- A dynamic model system that trains periodically when the model's accuracy drops for any environmental changes and should be deployed.

12. References

Agarwal, R. (2022, March 31). *Applications of Deep Learning: Convolutional Neural Network Models In the Healthcare Industry: Part…*. Medium. https://rishavagarwal1708.medium.com/applications-of-deep-learning-convolutional-neural-network-models-in-the-healthcare-industry-part-9ce482c5714c

Agarwal, R., Patel, S., Katiyar, S., & Nailwal, S. (2021). Malware Classification Using Automated Transmutation and CNN. *Advanced Computing and Intelligent Technologies*, 73–81. https://doi.org/10.1007/978-981-16-2164-2_6

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and. *ArXiv:1602.07360 [Cs]*. https://arxiv.org/abs/1602.07360

Khandelwal, V. (2020, August 18). *The Architecture and Implementation of VGG-16*. Medium. https://pub.towardsai.net/the-architecture-and-implementation-of-vgg-16-b050e5a5920b

Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2019, May 24). *Do ImageNet Classifiers Generalize to ImageNet?* Proceedings.mlr.press; PMLR. http://proceedings.mlr.press/v97/recht19a.html

Sudhakar, & Kumar, S. (2021). MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things. *Future Generation Computer Systems*, *125*, 334–351. https://doi.org/10.1016/j.future.2021.06.029

Tsang, S.-H. (2019, March 20). *Review: ResNet — Winner of ILSVRC 2015 (Image Classification, Localization, Detection)*. Medium. https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

Zou, B., Cao, C., Tao, F., & Wang, L. (2022). IMCLNet: A lightweight deep neural network for Image-based Malware Classification. *Journal of Information Security and Applications*, *70*, 103313. https://doi.org/10.1016/j.jisa.2022.103313