**Deep Malware: Malware Image Classification using Deep Learning**

# Abstract

The rapid advances in the field of communication and networks has increased the size and complexity of the network. Due to these reasons, many malwares are generated that create a challenge for systems to detect these malwares accurately. Moreover, the presence of malicious software (malware) with the aim of launching various malware files within the network cannot be ignored. Although, there are numerous efforts by the researchers to develop procedures for automatic classification of malware. The methods of manually analyzing malwares are very time-consuming. Lately, deep learning-based methods are being used for the classification of malware. In this paper, we present a rapid and accurate malwares classification based on different Convolutional Neural Network (CNN) architectures including a custom CNN as well as commodity off-the shelf CNN architectures such as VGG-16, ResNet-50, Inceptionv3 models. This has been demonstrated on benchmark datasets of Malimg dataset [1], which is consists of malware images that were obtained after conversion of Malware binaries.

# Introduction

Cybercrimes and their associated threats have risen significantly with the unexpected emergence of the linked digital world [2], like the (IoT) and Software Defined Networks (SDNs). The Internet of Things (IoT) is an interconnected network of digital equipment and systems that are referred to as 'things' [3]. They have sensors, processing chips, as well as other technology built in, enabling them to collect and conversation of data over the internet. Another purpose of IoT Networks, like manufacturing automation and security buildings, is really to boost the performance of cloud applications. Through the close of 2022, the quantity of IoT devices is expected to touch 50 billion [3]. As a consequence of this development, the cases of cyber-attacks have grown, and so has the danger associated with them. A result of this, businesses and organizations are looking for new solutions to protect individual and company information stored on edge devices. However, in the presence of tremendous threats, conventional IoT network security techniques have proven ineffective [4]. In 2017, for illustration, hackers utilized an IoT fish tank thermometers to steal sensitive data from casinos. According to the report of Symantec Internet Security Threat[5], over 2.4 million novel harmful kinds were generated in 2018. As a reaction, there was a boom in objective of strengthening NIDSs' capacity to detect new threats. As a consequence, new innovative methods for enhancing energy effectiveness of Network Intrusion Detection Systems in identifying assaults are required (NIDSs).

The NIDS monitors incoming traffic inside a network was applied to measure possible attacks and safeguard digital products [6]. It tries to protect software developers' 3 security requirements: confidentiality, availability, and integrity known as (CIA [6]. It was created with the goal of providing effective cyber security protection in operational systems. The primary goal using NIDSs has traditionally become to identify botnet and risks. NIDSs are classified into two groups: Signature-based cyberattacks try to match and also correlate arriving communication patterns to a repository of preset signatures from before disclosed intrusions [7]. They usually have high detection rate for previously recognized intrusions, but they miss new or updated attacks that aren't in the system. NIDSs has to be responsive to changing detection approaches, as hackers regularly change their assault theories and procedures to bypass existing security protocols. The current system for adjusting signatures to stay up with evolving threat vectors is insecure. Unusual case NIDSs attempt to solve the shortcomings of trademark NIDSs by utilizing advanced statistics methodologies that have enabled researchers to detect internet traffic behavioral trends. Anomaly detection may be done in a number of methods, such as employing knowledge of statistics and Machine Learning-based techniques [7]. Since they correlate attack behaviors instead of signatures, they may gain tremendous precision and Detection Rate (DR) for zero-day assaults [8]. Anomalies NIDSs, from the other side, have large False Alarm Rates (FARs) since they can classify any non-secure communication as an abnormality.

Current trademark NIDS have been demonstrated to be unsuccessful in detecting zero-day vulnerabilities as they traverse IoT networks [9]. This would be because of the absence of common attacks signatures inside the system's register. Many solutions, especially machine learning, have been devised and applied with varying degrees of success in order to avoid similar things from happening again. Machine Learning is indeed a cutting-edge technique that could really study and identify potentially dangerous trends through network, assisting in the identification of security issues [10]. Deep Learning has become an emerging new topic of machine learning which has demonstrated to be especially good at finding complicated statistics [11]. Its techniques are inspired by biological brain systems, that transmit information via a network of interconnected levels. Every unit has a mathematical encoder that converts inputs into outputs. Most of these techniques have hidden layers that can uncover even more sophisticated trends in internet activity. To comprehend these trends, cyberattack vectors are employed, that can be determined from a variety of parameters provided by network communication, like packet services, protocols, count/size, and indicators. Every assault style has its own identifying pattern, which would be described as a sequence of events that, when left unnoticed, can compromise network security.

To enhance their productivity, researchers designed and evaluated a range of machine learning and deep learning algorithms, which were typically combined with the information reduction strategy. Whereas these algorithms have given positive results by applying a set of evaluation metrics. Such models, however, are still ineffective in detecting malware through real-world Networks. The tendency in this area was to exceed tremendous outcomes for a particular dataset, instead of looking deeper into ML-based virus detection models. As a response, a large number of studies have been directed in a real-world setting. Whereas these tactics are problematic in practice since they are typically evaluated using just 1 dataset with such a consistent list of attributes which might not be useful to obtain or keep in a real Network communication stream, they are useful in theory. Moreover, due to the design of machine learning and deep learning, there is sometimes space for improvement in hyper-parameters whenever assigned to different data. The purpose of this project is to create a model that is self-sufficient of an array of benefits and trained it on the Malimg datasets (images dataset). The results of the research will be used to improve the design of the built-in CNN model.

# Literature Survey

Generally, three main approaches are popular for network security. Several triggers are set up in the first approach to identify network attacks, such as when a threshold value is reached. This kind of monitoring, notifies the management when the threshold limit is reached, but it does not protect the network from attacks. The second alternative is to prevent an attack by implementing defense mechanisms that prohibit the attack from occurring. However, this raises a problem if a legal operation is labeled illegitimate, resulting in a denial of service. The last approach of preventing an attack is to set up protection mechanism during the analysis of the attack and to prevent the attack in the future. In the last two approaches, the Malware detection system (MDS) is establishing the Malware prevention system (MPS) [12]. The IDS can be configured on two locations base on the source of information. Initially, the sensor can be fix on Host system (HMDS) or lastly can be setting up on the network (NMDS) [13][14]. Network Malware Detection Systems are primarily responsible for observing network traffic by evaluating various parameters such as protocol utilization, packet analysis, and IP address validation [13][14][15]. An MDS is an essential tool for network security, and its performance is usually determined by the accuracy with which it predicts valid and illegitimate activities.

Numerous ML and DL models had been showed significant accuracies but with their limitations and flaws. Due to the misclassification of attacks, ML models are usually vulnerable to them [16][17]. By inaccurately classify the data, ML model can enable the hacker to dodge the IDS that stand the organization on risk of undetectable attacks. Influential assaults, security violation attacks, and specificity attacks are the three main categories of offences [18][19]. In this paper, we give the latest assessment of machine learning algorithms addressed to that same challenge of malware and spyware recognition and characterization.

The ability to see malware as a colored picture is a significant step forward in the malware classification process. The concept of displaying malware as colored pictures was initially proposed by study [20]. They displayed malware like a gray-scale picture with in domain of [0, 255], with 0 representing dark black and 255 representing full white. They noticed that the photos they acquired included multiple portions, each of which indicated distinct details about the malware. For categorization, they employed GIST to calculate texture characteristics from malware pictures and K-nearest neighbors. When it comes to Deep learning, it has achieved significant progress in a variety of fields, including image identification, voice recognition, text classification, and others [21]. Syed et al. [22] suggested a malware variations identification approach based on malware specimen behavior; the key step was seeking evidence on malicious behavior sequences. The behavior of malware samples was acquired using their approach, which involved operating these in a simulated world.

For malware classification, EulGyuim et al. [23] proposed a chart similarity method. His process began by converting section data from Malware code into grey scale photos, which were then used to produce entropy charts. For chart resemblance, they employed the Strelkov entropy similarity measurement approach. Using 1000 samples spanning 50 distinct Malware families, they were able to reach a 97.9% similarity ratio. Their approach can process a huge proportion of unpacked Malware binary samples quickly. However, authors expressly stated in the constraints portion of their study that calculating similarity for packaged Malware binary examples using entropy charts is a challenging process.

Viruses, Worms, Trojans, and Backdoors are all types of malwares that have different capabilities. Based on the nature of variations, these categories are more divided into groups. To avoid being caught, malware authors use a variety of evasion techniques like dead-code implantation, subroutine rearrangement, and code substitution to generate variations of an existing malware group [24]. When we talk about Malware, we came to that it's a malicious software that is created with the goal of damaging the computer systems. Graphics based assessment of information security assaults has recently been used in research investigations [25]. The graphical inspection of secure shell (SSH) brute-force attack efforts, which were recognized by colors for the numerous irregularities found, as well as the specifics of User IDs and Internet-Protocol (IP) addresses [26], was one semi-automated approach. Visualization methods were also used to show a huge data packet in one go. These graphics depict the connections among packet headers, allowing security experts to focus in on specifics. Another study [27]employed image-based research to describe the timeline of a malicious assault like a phishing scam, with colors to showing which types of system connection was effective.

In context of malware categorization, Bensaoud et al. [28] utilized 6 deep learning algorithms. VGG16, Inception V3, and ResNet50 are prior champions of ISVLVRC challenge, while the remaining 3 algorithms are CNN-SVM, GRU-SVM, and MLP-SVM, which use Support Vector Machine (SVM) to improve neural networks. Researchers used the Malimg data for training all of the algorithms, and at the end findings show that the original Inception V3 model had the highest precision of 99.24 percent of all the research they performed. Naeem et al. [29] used a DCNN algorithm to transform APK data into color pictures. Also, on Leopard Mobile malware dataset3, the classifier was 97.81 percent accurate, while running on a windows dataset, it was 98.47 percent accurate. Mercaldo and Santone [30] suggested a framework for employing guided deep learning for the identification of dangerous samples in a definite way. Researchers used a collection of attributes extracted from colored photos to create various algorithms that could identify each virus category and variation within it. The group detection method had a 93.50 percent prediction performance, whereas category detection method had a 95.80 percent accuracy rate.

While standard machine learning approaches have certain drawbacks, Malware categorization relying on Malware pictures with deep learning had emerged as a viable alternative because that removes a significant amount of features engineering task. Malware categorization using deep learning had grown increasingly appealing in the last 2 years. In malware picture categorization, Yue [31] presented a deep CNN with scaled Softmax cost function. According to their findings, the new error rate may be used to adapt other common convolutional neural networks and increase classiﬁcation accuracy. Ni et al. [32] introduced the MCSC Malware classification system, which used sim-hash to transform deconstructed malware coding into grey pictures then using a convolutional neural network for identify their groups. Kalash et al. [33] also presented a CNN-based Malware categorization framework. This approach obtained great accuracy by utilizing VGG16's before-training algorithms.

Rezende et al. [34] applied the ResNet-50 design for spam filtering using deep learning too. The CNN layers of ResNet-50 pre-trained upon that ImageNet dataset were frozen to train the DNN. Gibert et al. [35] introduced a format independent deep learning strategy for Malware classification containing a collection of discriminating sequences generated from ransomware photos, inspired from the visual resemblance amongst malwares from the same groups. Bhodia et al. [36] compared photo transfer learning-based malware detection to KNN, a basic machine learning technique. In a virtual zero-day test, the findings revealed that the approach is relies on picture transfer learning outperformed KNN. In this study, we proposed the image-based classification of malwares. To achieve this, Malimg dataset was used to train CNN based models. The built-in CNN models were modified for Malimg dataset to achieve the efficiency and reliability.

# Methodology

Dataset: The proposed used the Malimg dataset for the classification of Malwares. The Malimg dataset was downloaded from Kaggle that is basically consists of 9458 malware samples which were split into 25 classes. The major feature of this dataset is that they are not providing malware samples once, but alternatively their images as they seem on disk. In a related way to the work in bytes of executable files are inconsequentially assigned to floats, which will later be elucidated as pixel values of the grayscale image. Unsurprisingly, the malware classes in the dataset are imbalanced and the majority class is ‘Allaple.A’ that contains 2949 samples, While the lowest class contains only 80 samples. The random samples of dataset are shown in Fig 1. It is clear that the images in each category have different styles that allow to distinguish between the samples of a family no matter what samples they are in another family.



Figure : Samples of Malimg dataset from different Malware Classes.

### Experimental setup: A hardware machine with 8GB RAM, 1TB HDD with 11G enable GPU was used for the experiments. The environment is configured with python version 3.6.10 for designing and testing the models. Numerous libraries were installed for utilities. The list of used libraries and their purpose is following

**Numpy:** NumPy refers for numerical python, a Python module that allows you to compute and manipulate many-dimensional and one-dimensional array items.

**Keras:** Keras is a Python-based deep learning API that lies on top of TensorFlow, which is machine learning platform. It was developed with the goal of permitting the rapid testing. It is developed with the aim to provide the results as soon as possible for doing good research.

**TensorFlow:** TensorFlow is a machine learning and artificial intelligence runtime environment that is 100% open-source and free to use. It can also be used for a variety of applications. But still, it focuses on deep neural network-based training and validation.

**Pillow:** Pillow is known as a Python Imaging Library (PIL) that permits you to browse, manipulate, and save images in Python. The updated iteration can recognize and handle a wide range of file categories. Writing support is limited to some of the most widely used exchange and presenting formats on purpose.

**Scikit-learn:** In Python, Scikit-learn (Sklearn) is the widely usable and powerful machine learning library. It uses a Python consistency API to give a set of fast tools for machine learning and statistical modelling, such as categorization, prediction, clustering, and data pre-processing.

**Pandas:** Pandas is indeed a data processing and analyzing software package for the Python programming language. It includes data formats and methods for processing numeric records and time-series data, especially. It's open-source software with a three-clauses BSD license.

**Seaborn:** Seaborn is indeed a Python module for creating statistical visuals. It is based on matplotlib & tightly interacts with panda’s dataset models. Seaborn assists you in exploring and comprehending your data. Its charting units operate with data frames and matrices comprising the whole data, doing necessary semantic mapping and statistical aggregation inside to plot useful graphs.

**Matplotlib:** Matplotlib is indeed a Python & its extension NumPy-based cross-platforms like that of data visualization and graphical charting package. Resultantly, it serves as an appropriate open-source replacement for MATLAB. The APIs (Application Programming Interfaces) of matplotlib may also be used to integrate graphs in graphical user interfaces.

**TQDM:** TQDM is indeed a Python module that allows you to create Progress charts or Progress Bars. The term TQDM is derived from an Arabic word Taqaddum, whose meaning is 'progress' in simple English.  TQDM can be easily implemented in our functions, loops, and even in Pandas libraries.

Pre-processing and Training:All the dataset images were passed from pre-processing pipeline to for suitable training of deep learning models. All the images were resized on the same scale to enable them for the training of the model. Next, the images were rescaled from RGB to Gray scale by the ratio of 1/255 to normalize the RGB values. if they are in range 0-255 the values are too high for good model performance. Lastly, the dataset was divided into two sets: training set and testing set. The Malware images were distributed in training and testing set with the ratio of 70% and 30% respectively. The 70% images of each class were assigned to training set and 30% to testing class rather than the collective distribution.

After the preparation of malware image dataset, several deep learning models were trained for the classification of Malwares. Initially the custom CNN was designed with convolutional, max pooling and flatten layers. The architecture of the model was based on the 2 convolutional layers, each followed by the MaxPooling layer, flatten layer and lastly two dense layers. The first convolutional layer performed as input layer and last dense layer used as output layer. The complete architecture of custom CNN model is shown in Fig 2. The Adam optimizer and categorical cross entropy is used as loss function. The rest of the hyper parameters were used with the default value. The feature of Reducing learning rate on plateau is used for the better performance of the classifier.

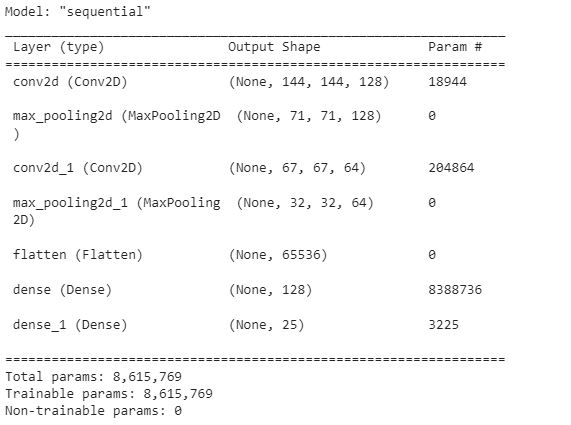
****

Figure : Architecture of proposed custom CNN for Malimg classification.

Further, three different deep learning models label as VGG-16, ResNet50, and InceptionV3 were trained for the classification of Malware images. All the models were trained with the Adam optimizer and cross entropy loss function. The models were trained for 50 epochs with early stopping mechanism. The result of all trained models was compared using the evaluation measures. Lastly, the model composition was used for the classification of malware images. Composite model was based on the integration of trained models. The custom CNN model, VGG16, ResNet50, and InceptionV3 were combined to predict a single result. The algorithm of composite model is following

Algorithm : Composite model algorithm for the classification of malwares.

|  |
| --- |
| **Input:** Sample image of Malimg dataset  **Output:** Class or Malware type of image. |
| 1. *Sample-image = load (“path of the testing set image)* 2. *Result1 = custom-CNN.* ***Predict*** *(Sample-image)* 3. *Result2 = VGG16.* ***Predict*** *(Sample-image)* 4. *Result3 = ResNet50.* ***Predict*** *(Sample-image)* 5. *Result4 = InceptionV3.* ***Predict*** *(Sample-image)* 6. *Final-Result =* ***max\**** *(Result1, Result2, Result3, Result4)* 7. ***Return*** *Final-Result* |
| \* Max function will take the list of all results and return a class value with maximum entrances. For instance, max (2, 4, 5, 2, 3, 4, 7, 4) will return the value 4. |

Evaluation Measures:To compare the performance of trained CNN models, different evaluation measures were used. The purpose of these evaluation measures to compute a numeric value of model in term of model performance by using different mathematical formulas. The selected evaluation measures for the proposed are accuracy, precision, recall and F1-score.

**Accuracy:** The simplest intuitive performance metric is accuracy, which is just the fraction of properly predicted observations to all observations. One would believe that if our model is accurate, it is the best. Yes, accuracy is a useful statistic, but only when the datasets are symmetric and the number of samples for each class are almost balance. Although the Malimg dataset is not a class balance dataset, but the accuracy of all trained models was calculated for fair comparison using Eq 1.

Eq. 1

**Precision:** Precision is the fraction of accurately predicted positive samples over the all predicted positive samples. The precision of trained models was calculated by the Eq 2.

Eq. 2

**Recall:** Recall is the fraction of truly predicted positive samples over the all-positive samples in actual class. The recall of the trained models was calculated by the Eq 3.

Eq. 3

**F1 Score:** F1 score is the weighted mean of true positive rate and false positive rate and it take both into account. The F1 score is calculated by the Eq 4.

Eq. 4

# Results and Discussion

This study proposed the deep learning model for the classification of malwares. For this purpose, the Kaggle malware dataset was used based on Malware images. The dataset was split into training and testing with the 70-30 ratio. The 70% ratio of Malimg dataset was used for training and the rest of 30% was used for testing. Several built-in CNN models were trained with the Malimg dataset including a custom convolutional neural network. The performance of each model was evaluated by the evaluation measures including accuracy, precision, recall and F1 score.

Firstly, the custom CNN model was trained that was based on convolutional layers, MaxPooling layers and dense layers. The Adam optimizer and Cross Entropy was used as with default hyper parameters. The CNN model showed the 90% accuracy. The accuracy and loss of CNN model during training is showed in fig 3 and 4 respectively. The confusion matrix of CNN model showed the class wise performance of the model (Fig 5). Next the three different built-in deep learning models (VGG16, ResNet50, and InceptionV3) were used for the classification of malwares.

|  |  |
| --- | --- |
| Figure 3: Accuracy of CNN model during training. | Figure 4: Loss of CNN model during training |
| Figure 5: Confusion matrix of CNN model | |

The VGG16 model was trained on the Malimg dataset for the classification of the malwares. VGG16 is a convolutional neural network (CNN) design that won the 2014 ILSVR(ImageNet) contest. It is regarded as being one of the best vision model architectures ever created. All across the design, the convolutional and max pool layers are arranged in the same way. The transfer learning technique was used for the training of the model. After the training of the model, the testing set was used for the evaluation of the model. Model showed the 80% accuracy for testing set. The accuracy and loss of the model with confusion matrix is also shown in Fig 6-7 and Fig 10 respectively. Next the ResNet50 was trained for classification purpose with the same hyper parameters. ResNet-50 is a 50-layers deep CNN model. You may import the pre-trained model versions of this network from the ImageNet database, which has been tested on over thousands of the photos. This network can categories photos into 1000 different object classes, including keyboards, mice, pencils, and a variety of creatures. ResNet50 showed the 81% accuracy for testing data. The training and validation accuracy and loss is also shown in Fig 8-9. For the calculation of precision and recall, the confusion matrix of ResNet50 model was plotted and showed in Fig 11.

|  |  |
| --- | --- |
| Figure 6: Accuracy of VGG16 model for training and validation set. | Figure 7: Loss of VGG16 model for training and validation set. |
| Figure 8: Accuracy of ResNet50 model for training and validation set. | Figure 9: Loss of ResNet50 model for training and validation set. |
| Figure 10: Confusion matrix of VGG16 model for testing set. | Figure 11: Confusion matrix of ResNet50 model for testing set. |

The 3rd built-in model for the classification of malware was InceptionV3. Inception-v3 is a CNN model design which relates to Inception family that includes Label Smoothing, factorized 7 x 7 convolutions, and the inclusion of an extra classification algorithm to transport labelled data deeper down the structure, among the other enhancements (In addition, batch normalization for layers in the side-head is used). InceptionV3 showed the 87% accuracy for testing set while the training and validation accuracy is shown in Fig 12. The loss of InceptionV3 model during the training of the model is also represented in Fig 13 with CM in Fig 14. Lastly, we prepared a composite model by integrating the all-trained models. The results were compiled by getting the prediction from all models and the final decisions were made base on the majority rule. The composite model showed the 92% accuracy. The confusion matrix of composite model for testing data is shown in Fig 15.

|  |  |
| --- | --- |
| Figure 12:Accuracy of InceptionV3 model for training and validation set. | Figure 13: Loss of InceptionV3 model for training and validation set. |
| Figure 14:Confusion matrix of InceptionV3 model for testing set. | Figure 15: Confusion matrix of composite model for testing set. |

After the training of all the models, the results were compared by the evaluation measure. As the original dataset is not balance, but the testing set had the approximately equal number of malware images for each class. Hence, the accuracy is good measure for balance dataset. Resultantly, all the models were compared against accuracy score to evaluate the best model for Malware classification. CNN showed the 90% accuracy score for malware classification. Confusion metric showed that majority of classes are 100% accurately classified. However, there are 8 classes in confusion matrix (CM) that have the 90 to 95% accuracy score (Fig 5). Yuner A Malware type is the prominent class that downscale the accuracy of custom CNN model. CM showed that all the samples of Yuner A malware class are wrongly classified as Autorun K class. Resultantly, the accuracy of Yuner A malware type was 0%. This behaviour of Yuner A class raises different probabilistic issues and confirmed that custom CNN is unable to train on the samples of Yuner A class. Probably the pattern of Yuner A class is too much complex for the learning of model or there may be no pattern exist for learning. However, the model showed the significant result for the classification of malware types other than the Yuner A type.

Next, the trained weights were used for the training of VGG16 model and model showed the 80% accuracy for testing set. Several malware classes are accurately learned by the VGG16 model and it showed the 100% accuracy for these classes (Fig 10). There are seven other classes that showed the 80 to 90% accuracy. In Fig 10, CM showed that the most misclassified types of malwares are Autorun K and skintrim N for VGG16 model. All the samples in testing test belonging to Autorun K malware are wrongly classified as Yuner A malware type. On the other hand, the five sample of skintrim N are classified as Alueron and ten samples are classified as Instantaccess. Collectively, the 15 samples of Autorun K type are misclassified out of 18 samples. Rest of the malware classified are classified on average 90% accuracy. However, with the Autorun K and skintrim class VGG16 model showed the 80% accuracy.

ResNet50 model was also trained on Malimg dataset for the classification of malware images. The pretrained weights of ImageNet were used for the training of ResNet50 model. Model showed the 81% accuracy for testing set. The confusion matric of ResNet50 showed that the majority malware types are accurately learn and classified with 100% accuracy score (Fig 11). In the CM of ResNet50, only 5 samples are misclassified belonging to three different classes. However, there are few classes are completely misclassified and showed that model was unable to learn the pattern of these images for classification. Yuner A and Autorun K are completely misclassified as third malware type label as Fakerean. The Obfuscator malware type was completely misclassified as Instantaccess malware type. Except of these three classes, all the remaining classes are classified with approximately 99% accuracy score. IceptionV3 also train for the classification of malware types. InceptionV3 showed the similar result as custom CNN. It also had the majority classes with 90 to 95% accuracy and the Yuner A class was completely misclassified as Autorun K malware type as similar to custom CNN (Fig 14). However, the overall accuracy score of InceptionV3 was less then the custom CNN accuracy score.

Lastly, a composite model was used for the classification the malware types. Composite model integrated the all described model and take the decision on majority bases. It does not predict itself and accept a list of prediction for decision. The composite model returns back the class that have the maximum instances in the given list. Composite model showed the 92% accuracy for testing data. The confusion matrix of composite model showed that the majority of the malware types are 100% accurately classified (Fig 15). But there was again Yuner A class that completely misclassified. All the 15 samples of Yuner A malware types were again classified as Autorun K class by composite model. However, there are 9 classes that have the accuracy score more than 95% and the average accuracy score for all malware types except Yuner A was approximately 98% for composite model.

As the results of all models showed that there are some malware types that are too much difficult to learn for all models. The Yuner A and Autorun K malware types seems most similar malware types, as these classes are interchangeably misclassified by different models. Custom CNN and InceptionV3 classified all the Yuner A samples as Autorun K malware type while the VGG16 completely misclassified all the samples of Autorun K malware type as Yuner A malware type. On the other hand, ResNet50 misclassified the all samples of both classes as third malware type (Instantaccess). Collectively, there were the three classes (Autorun K, Instantaccess, Yuner A) that downscaled the performance of trained models. The composite model improves the performance by correctly classified the Autorun K and Instantaccess class. However, the Yuner A is still problematic class for models, as it also misclassified by the composite model. As the composite model take decision on majority instances and majority models classified the Yuner A malware type as Autorun K malware type. It may be due to the similarity of Yuner A class with autorun and Instantaccess class or may be the patter of Yuner A class too much difficult for the learning of the model. There is need to develop a complex model that deal with the complexity of Yuner A malware images in future studies. There may be two models that perform sequentially to classify malware image. Firstly, the problematic classes are merged and label as one parent class. The model A may classify between the simple class and parent class and model B classify the image into sub type of malware. Collectively, there is need t update the model for complex class (Yuner A). The existing models showed significant result for malware classification to perform in real world environment.

# Conclusion

In the proposed study, we use the Malimg dataset (image base dataset of malwares) for the classification of malwares. Further, several built-in deep learning models were trained for the fair comparison of the models. Lastly, the trained models were integrated to proposed a composite model. Few similar classes like Yuner A, Instantaccess and Autorun K downscale the performance of the model. As the all-built-in models continuously misclassify the sample belonging to these three classes, it is assumed that these classes are most similar and models and unable to distinguished them. However, the composite model fixed the two classes prediction (Instantaccess and Autorun K) but the Yuner A class still misclassified by the composite model. The accurately prediction of Yuner A class of malwares need the complex architecture of deep learning model or may need required few images pre-processing steps. Moreover, our composite model classifies the malwares with 92% accuracy. The significant accuracy score concludes that with the image base technique, model is robust enough to accurately classify the malwares. However, for the accurate classification of complex malware types, there is need to modify the training scheme or model architecture in future studies.

# References

[1] “Microsoft Malware Classification Challenge (BIG 2015) | Kaggle.” https://www.kaggle.com/c/malware-classification/data (accessed Jan. 21, 2022).

[2] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, “A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services,” *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 3453–3495, Oct. 2018, doi: 10.1109/COMST.2018.2855563.

[3] M. A. Khan and K. Salah, “IoT security: Review, blockchain solutions, and open challenges,” *Future Generation Computer Systems*, vol. 82, pp. 395–411, May 2018, doi: 10.1016/J.FUTURE.2017.11.022.

[4] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, “Internet of Things (IoT): Taxonomy of security attacks,” *2016 3rd International Conference on Electronic Design, ICED 2016*, pp. 321–326, Jan. 2017, doi: 10.1109/ICED.2016.7804660.

[5] “ISTR Internet Security Threat Report Volume 24 |,” 2019.

[6] F. Y. Sattarova, “Integrating intrusion detection system and data mining,” *Proceedings - 2008 International Symposium on Ubiquitous Multimedia Computing, UMC 2008*, pp. 256–259, 2008, doi: 10.1109/UMC.2008.59.

[7] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Computers & Security*, vol. 28, no. 1–2, pp. 18–28, Feb. 2009, doi: 10.1016/J.COSE.2008.08.003.

[8] P. Amoli, T. Hamalainen, … G. D.-… (International J. of, and undefined 2016, “Unsupervised network intrusion detection systems for zero-day fast-spreading attacks and botnets,” *users.jyu.fi*, Accessed: Oct. 07, 2021. [Online]. Available: http://users.jyu.fi/~pavahdan/Unsupervised%20Network%20Intrusion%20Detection%20Systems%20for%20Zero-Day%20Fast-Spreading%20Attacks%20and%20Botnets.pdf

[9] M. J. Hashemi, G. Cusack, and E. Keller, “Towards evaluation of NIDSs in adversarial setting,” *Big-DAMA 2019 - Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks, Part of CoNEXT 2019*, pp. 14–21, Dec. 2019, doi: 10.1145/3359992.3366642.

[10] C. Sinclair, L. Pierce, and S. Matzner, “An application of machine learning to network intrusion detection,” *Proceedings - Annual Computer Security Applications Conference, ACSAC*, vol. Part F133431, pp. 371–377, 1999, doi: 10.1109/CSAC.1999.816048.

[11] Q. Niyaz, W. Sun, A. Y. Javaid, and M. Alam, “A deep learning approach for network intrusion detection system,” *EAI International Conference on Bio-inspired Information and Communications Technologies (BICT)*, 2015, doi: 10.4108/EAI.3-12-2015.2262516.

[12] O. Ibitoye, O. Shafiq, A. M.-2019 I. Global, and undefined 2019, “Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks,” *ieeexplore.ieee.org*.

[13] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, “From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods,” *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 3369–3388, Oct. 2018, doi: 10.1109/COMST.2018.2854724.

[14] N. T. Van, T. N. Thinh, and L. T. Sach, “An anomaly-based network intrusion detection system using Deep learning,” *Proceedings - 2017 International Conference on System Science and Engineering, ICSSE 2017*, pp. 210–214, Sep. 2017, doi: 10.1109/ICSSE.2017.8030867.

[15] E. Benkhelifa, T. Welsh, and W. Hamouda, “A critical review of practices and challenges in intrusion detection systems for IoT: Toward universal and resilient systems,” *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 3496–3509, Oct. 2018, doi: 10.1109/COMST.2018.2844742.

[16] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial Machine Learning at Scale,” *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, Nov. 2016.

[17] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, “A taxonomy and survey of attacks against machine learning,” *Computer Science Review*, vol. 34, p. 100199, Nov. 2019, doi: 10.1016/J.COSREV.2019.100199.

[18] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, “Adversarial Machine Learning \*,” *Proceedings of the 4th ACM workshop on Security and artificial intelligence - AISec ’11*, 2011, doi: 10.1145/2046684.

[19] H. Xiao, H. Xiao, and C. Eckert, “Adversarial Label Flips Attack on Support Vector Machines,” *Frontiers in Artificial Intelligence and Applications*, vol. 242, pp. 870–875, 2012, doi: 10.3233/978-1-61499-098-7-870.

[20] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware Images: Visualization and Automatic Classification,” *Proceedings of the 8th International Symposium on Visualization for Cyber Security - VizSec ’11*, 2011, doi: 10.1145/2016904.

[21] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature 2015 521:7553*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[22] S. Z. M. Shaid and M. A. Maarof, “Malware Behaviour Visualization,” *Jurnal Teknologi*, vol. 70, no. 5, pp. 25–33, Sep. 2014, doi: 10.11113/JT.V70.3512.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[24] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, “Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey,” *Information Security Technical Report*, vol. 14, no. 1, pp. 16–29, Feb. 2009, doi: 10.1016/J.ISTR.2009.03.003.

[25] H. Shiravi, A. Shiravi, and A. A. Ghorbani, “A survey of visualization systems for network security,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1313–1329, 2012, doi: 10.1109/TVCG.2011.144.

[26] “Security Data Visualization | SANS Institute.” https://www.sans.org/white-papers/36387/ (accessed Jan. 21, 2022).

[27] W. Shanks, A. : Angel, and A. Parrizas, “Global Information Assurance Certification Paper Enhancing Intrusion Analysis through Data Visualization GIAC (GGIA) Gold Certification”, Accessed: Jan. 21, 2022. [Online]. Available: http://www.giac.org/registration/gcia

[28] A. Bensaoud, N. Abudawaood, and J. Kalita, “Classifying Malware Images with Convolutional Neural Network Models,” *International Journal of Network Security*, vol. 22, no. 6, pp. 1022–1031, Nov. 2020, doi: 10.6633/IJNS.202011\_22(6).17.

[29] H. Naeem *et al.*, “Malware detection in industrial internet of things based on hybrid image visualization and deep learning model,” *Ad Hoc Networks*, vol. 105, p. 102154, Aug. 2020, doi: 10.1016/J.ADHOC.2020.102154.

[30] F. Mercaldo, A. S.-J. of C. V. and Hacking, and undefined 2020, “Deep learning for image-based mobile malware detection,” *Springer*, vol. 16, pp. 157–171, 2020, doi: 10.1007/s11416-019-00346-7.

[31] S. Yue, “Imbalanced Malware Images Classification: a CNN based Approach,” Aug. 2017, Accessed: Jan. 21, 2022. [Online]. Available: https://arxiv.org/abs/1708.08042v1

[32] S. Ni, Q. Qian, and R. Zhang, “Malware identification using visualization images and deep learning,” *Computers & Security*, vol. 77, pp. 871–885, Aug. 2018, doi: 10.1016/J.COSE.2018.04.005.

[33] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, “Malware Classification with Deep Convolutional Neural Networks,” *2018 9th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2018 - Proceedings*, vol. 2018-January, pp. 1–5, Mar. 2018, doi: 10.1109/NTMS.2018.8328749.

[34] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. de Geus, “Malicious software classification using transfer learning of ResNet-50 deep neural network,” *Proceedings - 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017*, vol. 2017-December, pp. 1011–1014, 2017, doi: 10.1109/ICMLA.2017.00-19.

[35] D. Gibert, C. Mateu, J. Planes, and R. Vicens, “Using convolutional neural networks for classification of malware represented as images,” *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 15–28, Mar. 2019, doi: 10.1007/S11416-018-0323-0/FIGURES/6.

[36] N. Bhodia, P. Prajapati, F. di Troia, and M. Stamp, “Transfer Learning for Image-Based Malware Classification,” *ICISSP 2019 - Proceedings of the 5th International Conference on Information Systems Security and Privacy*, pp. 719–726, Jan. 2019, doi: 10.5220/0007701407190726.