EEMDS: Efficient and Effective Malware Detection System with Hybrid Model based on XceptionCNN and LightGBM Algorithm

[1,2*]Monday Onoja, [2,3]Abayomi Jegede, [2]Nachamada Blamah, [4]Olawale Victor Abimbola and [1]Temidayo Oluwatosin Omotehinwa

[1]Department of Mathematics and Computer Scicence, Faculty of Science, Federal University of Health Sciences, P.M.B 145, Otukpo, Nigeria.
[2]Department of Computer Scicence, Faculty of Natural Science, University of Jos, P.M.B 2084 Jos, Nigeria.
[3]Africa Centre of Excellence on Technology Enhanced Learning, National Open University, Abuja, Nigeria.
[4]Creative Advanced Technologies, Dubai, UAE.

email: [1,2*]monday.onoja@fuhso.edu.ng; [2,3]jegedea@unijos.edu.ng; [2]blamahn@unijos.edu.ng
[4]abimbolaolawale41@gmail.com; [1]oluomotehinwa@gmail.com

*****Corresponding author*

Received: 13 June 2022 | Accepted: 24 October 2022 | Early access: 28 October 2022

Abstract - *The security threats posed by malware make it imperative to build a model for efficient and effective classification of malware based on its family, irrespective of the variant. Preliminary experiments carried out demonstrate the suitability of the generic LightGBM algorithm for Windows malware as well as its effectiveness and efficiency in terms of detection accuracy, training accuracy, prediction time and training time. The prediction time of the generic LightGBM is 0.08s for binary class and 0.40s for multi-class on the Malimg dataset. The classification accuracy of the generic LightGBM is 99% True Positive Rate (TPR). Its training accuracy is 99.80% for binary class and 96.87% for multi-class, while the training time is 179.51s and 2224.77s for binary and multi classification respectively. The performance of the generic LightGBM leaves room for improvement, hence, the need to improve the classification accuracy and training accuracy of the model for effective decision making and to reduce the prediction time and training time for efficiency. It is also imperative to improve the performance and accuracy for effectiveness on larger samples. The goal is to enhance the detection accuracy and reduce the prediction time. The reduction in prediction time provides early detection of malware before it damages files stored in computer systems. Performance evaluation based on Malimg dataset demonstrates the effectiveness and efficiency of the hybrid model. The proposed model is a hybrid model which integrates XceptionCNN with LightGBM algorithm for Windows Malware classification on google colab environment. It uses the Malimg malware dataset which is a benchmark dataset for Windows malware image classification. It contains 9,339 Malware samples, structured as grayscale images, consisting of 25 families and 1,042 Windows benign executable files extracted from Windows environments. The proposed XceptionCNN-LightGBM technique provides improved classification accuracy of 100% TPR, with an overall reduction in the prediction time of 0.08s and 0.37s for binary and multi-class respectively. These are lower than the prediction time for the generic LightGBM which is 0.08s for binary class and 0.40s for multi-class, with an improved 100% classification accuracy. The training accuracy increased to 99.85% for binary classification and 97.40% for multi classification, with reduction in the training time of 29.97s for binary classification and 447.75s for multi classification. These are also lower than the training times for the generic LightGBM model, which are 179.51s and 2224.77s for the binary and multi classification respectively. This significant reduction in the training time makes it possible for the model to converge quickly and train a large sum of data within a relatively short period of time. Overall, the reduction in detection time and improvement in detection accuracy will minimize damages to files stored in computer systems in the event of malware attack.*

Keywords: Anomaly-based Detection, LightGBM, Machine Learning, Malware Detection, XceptionCNN.

1 Introduction

Malware, also known as malicious software can be described as any instruction set that is compromised to alter a computer system and impose harm on users and organizations (Abusitta, 2021). A malware is categorized based on its activities and execution process (Singh & Singh, 2020). The internet has made great contributions in communication, however, it has consequently led to the rise in malware distribution. The developments of web services with increasing speed have made productive advantage available to end-users. The number of people using the Internet was about two billion in 2010 (Chang et al., 2013). A report from Dasient, and cited by Chang et al. (2013), suggests that the number of malware delivering websites doubled between 2009 and 2010. "There were 3.424 billion people using the Internet by July 2018" (Wang, 2018). Carrying out activities on an infected website is a sufficient pathway for an attacker to take advantage of the weakness of a browser.

Malware analysis is essential in order to build successful malware detection techniques. The focus of malware analysis is to understand the intent and activities of malware (Wong et al., 2021). Malware analysis may be static, dynamic, or hybrid depending on the way and manner it is carried out. Analysts use static analysis, dynamic analysis or a combination of both methods (hybrid) to understand and explain the mode of operation of malware and the effects on the system (Wong et al., 2021).

Malware detection is the process of recognizing malicious sets of instruction from benign ones, so that a defense can be built, in order that the system can be protected or recovered from any malicious effects (Landage & Wankhade, 2013). Malware detection techniques identify malicious codes and prevent the system from its effect and possible loss of information. A malware detector uses malware detection techniques to identify activities of malware. Figure 1 shows malware detection techniques and approach as presented by Kumar (2017).

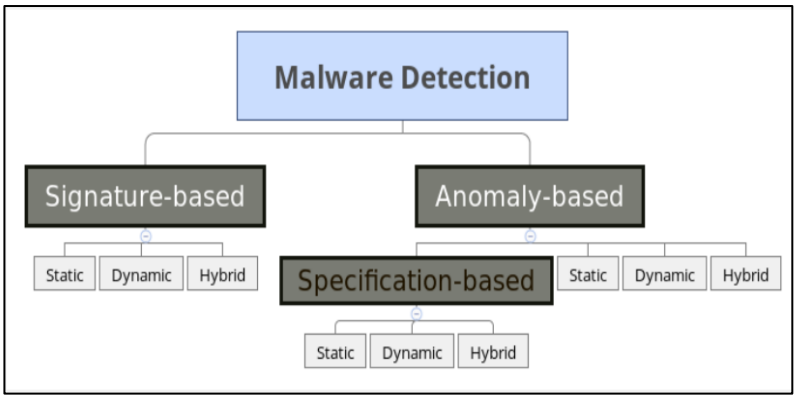

Figure 1: Malware detection techniques (Kumar, 2017)

1.1 Signature-Based Detection

A signature is a chain of information that describes the activities of a particular malware (Damodaran et al., 2017). In Signature-Based Detection approach, unique signatures are detached from captured malware files. The signatures are further used to detect malware with similar characteristics. Signature-based approach is suitable for generic malware which do go through significant behavioural modification (Abusitta, 2021). However, attackers easily manipulate malware signatures in order not to be detected by antivirus software (Abusitta, 2021). Signature-based models are very effective in known malware detections, but, it is unable to identify new ones (Bazrafshan et al., 2013).

1.2 Anomaly-Based Detection

In Anomaly-based detection; the behaviors of malware during runtime are studied in a training phase, after which the executable is tagged as malicious or benign during testing phase based on extracted patterns in the training phase (Damodaran et al., 2017). Behavior-based method is capable of detecting new and unknown malware and malware that uses obfuscation techniques. The main limitations of the behavior-based detection are: a substantial False Positive Rate (FPR) and unnecessary testing time (Bazrafshan et al., 2013).

Malware issue has developed into a serious issue in computing. According to Gibert et al. (2020), machine learning technique is the best technique that is needed to protect a computer system due to rise in malware attack. Using malware images makes malware classification easier (Pant & Bista, 2021). Image-based techniques are robust against many types of obfuscations (Bhodia et al., 2019). Omitting irrelevant features fasten and make algorithm to perform better (Şahin et al., 2021). While LightGBM technique is the best of the Gradient Boosting Decision Tree Algorithms (GBDT) and has demonstrated its suitability for malware detection (Abbadi et al., & Pan et al., 2020), XceptionCNN is an effective and less complex neural network for robust feature extraction (Shaheed & Zhang, 2022).

We conducted a preliminary study which reveals the prediction time of the generic LightGBM to be 0.08s for binary class and 0.40s for multi-class on the Malimg dataset with 10,381 malware samples. The preliminary study further reveals a classification accuracy of 99% (TPR), with training accuracy of 99.80% for binary classification on Malimg dataset and 96.87% for multi classification on the same malware samples. A growth in data size may lead to corresponding increase in time of prediction. Although the classification accuracy obtained from our preliminary experiment seems to be good, it could be further improved in order to enforce the effectiveness of the algorithm. The prediction time, performance accuracy, and training time obtained from the preliminary experiment also leave room for improvement. Hence, there will be a need to make the classification accuracy of the model better for effective decision making, reduce the prediction time for efficiency, and improve the performance and accuracy for effectiveness on larger samples. Our study, which hybridized XceptionCNN with LightGBM has the following contributions and novelty:

- Improved training accuracy of the model for effective decision making.
- Reduction in the detection time and improvement in detection accuracy, which will minimize damages to files stored in computer systems in the event of malware attack.
- Reduction in the training time, which will enable the model to converge quickly and train a large amount of data within a relatively short period of time.
- The proposed model can detect both known and new malware variants.
- The training accuracy of the proposed model is higher than those of the existing models.
- This study is the first to compute and improve the detection time of the LightGBM algorithm

This study proposes a hybrid model based on LightGBM and XceptionCNN algorithms. The aim is to improve the efficiency and effectiveness of Windows malware detection. Our preliminary study revealed that the LightGBM technique which is the best of the GBDT algorithm, has proven to be suitable for Windows malware detection (Abbadi et al., 2020; Pan et al., 2020) and can be improved for effective and efficient malware detection. ML-based classifiers use underlying features to distinguish between malicious and benign applications, and detecting changes in those features when malicious modifies itself. Malware possess certain features which Machine learning algorithm can learn and use to predict if an executable file is a malware or benign sample. Such that, if such an executable file behaves in a certain way or attempts to modify the system or access privilege instructions, they may be classified as malware or benign based on their activities. Our technique can take a number of these features as input, learn the properties of these features, then, build a model for prediction of new samples. Using the mathematical function $f: m \rightarrow x$, where m is the given malware and x is their corresponding malware family, the model can also detect new variants of malware. This study will contribute to advances in malware detection. The proposed solution can be applied to similar studies in future.

The rest of this paper is organized as follows: Section 2 presents related work in malware detection, while Section 3 discusses experiments and implementation details. Section 4 covers the findings of our study and the implications of those findings, while Section 5 summarizes the study and draws conclusions based on the achieved objectives.

2 Related Work

Ke et al. (2017) posited that GBDT is one of the machine learning classifiers which is extensively utilized as a result of its effectiveness. LightGBM is a new GBDT algorithm with Gradient-based One-Side Sampling (GOSS) and the Exclusive Feature Bundling (EFB) (Ke et al., 2017). The GOSS and EFB makes LightGBM considerably effective than eXtreme Gradient Boosting (XGBoost) with regards to speed and memory usage (Ke et al., 2017). In an attempt to study user's click for click fraud detection, Minastireanu and Mesnita (2019) used the experimental test for LightGBM using a public dataset available on Kaggle. Their results on the GBDT Algorithm achieved 98% accuracy. In a different study, using the N-Gram model, Venkat et al. (2020) proposed a medicine approval system. In an attempt to boost the efficiency of the medicine approval system, a LightGBM model was

used to carry out medication examination. Ju et al. (2019) observed that single-convolution model was ineffective for wind power prediction; hence, they proposed a solution which integrates LightGBM algorithm with convolutional model to enhance the prediction accuracy and reliability. A study by Fonseca et al. (2017) showed that training LightGBM algorithm is faster than training XGBoost. The study did not compare the algorithms based on their classification time. In further study, using dataset made available through Kaggle's competition, Machado et al. (2019) evaluated the accuracy of two GBDT Models: XGBoost and LightGBM algorithm in predicting credit card customers' reliability status. The study assessed customer loyalty prediction accuracy through Root Mean Square Error and found that LightGBM achieved better than XGBoost. LightGBM-based method performed better than most generic methods (such as Support Vector Machine, XGBoost, or Random Forest) when applied fraud detection (Huang, 2020). Sun et al. (2020) also used LightGBM to combine the daily data of 42 kinds of primary crypto-currencies with key important pointers in order to predict the prices of crypto-currencies and obtain relevant information about the market. Their experimental results show that the robustness of the LightGBM model is better than the other models. The time complexity for the LightGBM is calculated as O(#Data x #Features) (Meidan et al., 2018). A malware classification approach converted malware binaries to grayscale images before using a trained CNN to build a model for classifying malware according to its family (Kalash et al., 2018). A deep learning architecture applied to Malimg malware dataset and Microsoft dataset has performance accuracy of 98.52% for malign dataset and 99.97% accuracy for Microsoft datasets (Kalash et al., 2018). In a similar study carried out by Bhodia et al. (2019), their deep learning architecture which was also experimented on the Malimg malware dataset yielded a training accuracy of 98.39% for binary classification and 94.80% for multi-classification. A study by Lo et al. (2019) classified malware into families based on the integration of deep CNN with Xception model. Experimental results show that the training accuracy of the Xception model on the Malimg datasets is 99.37%. The study did not evaluate the classification accuracy and prediction time of the model. Hussain (2019) proposed a hybrid technique based on gradient boosting classifier. The method used information such as target of applications, privileges, static data and dynamic data to detect malicious application. The approach has a good detection accuracy of 96%. This result still leaves room for improvement. In another study, Nawaz et al. (2021) performed hybrid analysis using Android application features which include permissions, targets, and network features. The study extracts permissions and intent from a suspected file. It also obtains network related information from java files. The use of Info Gain as a feature selection method results in precision of 0.99. The study did not apply their model to Windows malware domain. An improved solution uses a small set of highly discriminant features for automated malware detection (Fang et al., 2019). The goal is to address the limitations of classical feature selection techniques. DQFSA interacts with the feature space and uses Q-learning to train an agent in order to achieve high accuracy. The proposed approach performs better than existing baseline feature selection methods for malware detection using small feature sets. The study did not apply the framework to other selection tasks. In an attempt to construct a detection framework, Chen et al. (2020) used the characteristics of data and features of the attention mechanism to construct a sliding local attention mechanism model (SLAM). The performance accuracy of the proposed model is 0.9723. The study did explore the used of the technique for malware detection. Bensaoud and Kalita (2022) proposed a novel deep learning method for classifying malware images for effective and efficient malware detection. Experimental results based on about 100,000 benign and malicious PE, APK, Mach-o, and ELF show that the method has the highest accuracy of more than 99.87%. The method is also effective at detecting different malware evasion techniques. The detection time of the model was not considered. Pan et al. (2020) used Logistic Regression, KNN and LightGBM to build models based on datasets of heartbeat and threat reports. The results obtained from the respective models show that LightGBM has the highest accuracy with AUC of 0.720687. The study attempted to enhance the training accuracy of the models, however, the detection time and detection accuracy were not considered. Using a custom model based on convolutional neural network with a benchmark dataset (Malimg dataset), Pant and Bista (2021) achieved 99.64% training accuracy while classifying malware into their respective families. The study did not consider the detection time of the model. The study also did not evaluate the model based on binary classification of the malware. A preliminary study conducted by the authors of this article reveal that the LightGBM achieves 0.08s prediction time for binary classification on Malimg dataset with 10,381 malware samples and a prediction time of 0.40s for multi classification on the same malware samples. The preliminary experiments reveal that the classification accuracy of the generic LightGBM machine learning algorithm is 99% TPR. From the literature review, no study has deemed it fit to improve the classification accuracy or reduce the prediction time of the LightGBM algorithm for windows malware detection.

3 Methodology

The research process flow in Figure 2 depicts the sequence of activities required to accomplish the overall objectives of the study.

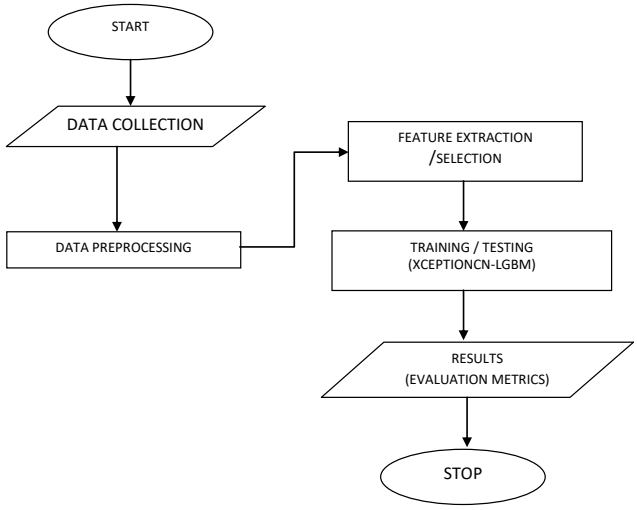

Figure 2: Research process flow

3.1 Data Collection

We used the Malimg malware dataset which contain 9,339 of Malware samples structured as grayscale images consisting of 25 malware families. Each of the malware families is made up of varying number of samples across the dataset. Malimg dataset is one of the most commonly used datasets for malware findings. The Malimg dataset was created by reading malware binaries into an 8-bit unsigned integer consisting of a matrix $M \in R^{m \times n}$ (Nataraj et al., 2011). The matrix could be seen as image (grayscale) having values within the range of [0, 255], where 0 represents black, 1 represents white. In addition, benign dataset used are 1,042 windows benign executable files extracted from windows environment and further converted into images. Table 1 presents a breakdown of the Malimg dataset into families and their variants.

Table 1: The Malimg Dataset (Nataraj et al., 2011).

NO.	Family Name	Family	Samples
1	Adialer.C	dialer	122
2	Agent.FYI	Backdoor	116
3	Allaple.A	worm	2,949
4	Allaple.L	worm	1,591
5	Alueron.gen!J	Trojan	198
6	Autorun.K	worm	106
7	C2LOP.gen!g	Trojan	200
8	C2LOP.P	Trojan	146
9	Dialplatform.B	Dialer	177
10	Dontovo.A	Trojan Downloader	162
11	Fakerean	Rogue	381
12	Instantaccess	Dialer	431
13	Lolyda.AA1	PWS	213
14	Lolyda.AA2	PWS	184
15	Lolyda.AA3	PWS	123
16	Lolyda.AT	PWS	159
17	Malex.gen!J	Trojan	136
18	Obfuscator.AD	Trojan Downloader	142
19	Rbot!gen	Backdoor	158
20	Skintrim.N	Trojan	80

21	Swizzor.gen!E	Trojan Downloader	128
22	Swizzor.gen!I	Trojan Downloader	132
23	VB.AT	worm	408
24	Wintrim.BX	Trojan Downloader	97
25	Yuner.A	worm	800
	Total	—	**9,339**

The analysis of the families and variants of the Malimg Dataset with a total of 9,339 dataset (Nataraj et al., 2011).

3.2 Data Preprocessing

The Malimg dataset used consists of images, hence, there was no preprocessing done on the dataset. We directly passed the images that consist of the dataset into XceptionCNN for automatic feature extraction. Thereafter, we saved the extracted features as CSV and passed it for classification and training. We also separated the dataset into training and testing sets. Since our model require images as input, we further converted the windows benign executable file to images using 'exe2image' converter, a digital image converter software available on github (Malith, n.d.), operated on windows.

3.3 Feature Extraction

A total of 10,381 images of two (2) classes were used for the binary classification and the same number of images of twenty (26) classes were used for the multi-classification. We applied an Advanced Convolutional Neural Network model (Xception) which extracted the image features with distinctive pattern automatically from the dataset, and the extracted features were saved as CSV files. Malware that share the same family are very similar in layout and image (Nataraj et al., 2011).

3.4 Training and Testing

We trained the models on 100 epochs using the following hyperparameters : learning rate of 0.03 & Max-dept of 10. We used a total of 10,381 data samples. We randomly selected 80% and 20% of the samples in each of the families for training and testing respectively, resulting in 8,304 and 2,077samples used for training and testing respectively. The LightGBM was used for training and testing.

3.5 LightGBM Classifier

LightGBM is a new gradient boosting framework. It is a decision tree algorithm which supports many algorithms like Gradient Boosting Machine (GBM), Gradient Boosting Decision Tree (GBDT), Gradient Boosted Regression Tree (GBRT), and Multiple Additive Regression Tree (MART). It is has high level scalability, precision, and efficiency (Ke et al., 2017). It is suitable for classification and other machine learning activities (Abbadi et al., 2020). It applies a leaf-wise splitting of the tree based on the best fit, unlike other boosting algorithms which use depth-wise or level-wise splitting. The leaf-wise growth of LightGBM reduces the level of loss, which results in faster speed and higher accuracy than other boosting algorithms. Figure 3 shows the leaf-wise tree growth structure of the LightGBM algorithm.

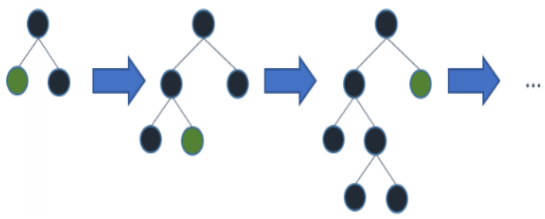

Figure 3: LightGBM Architecture (Khandelwal, 2017)

Figure 3 shows the leaf-wise tree growth (architecture) of the LightGBM algorithm as presented by Khandelwal (2017).

Leaf-wise splits results in high complexity and overfitting, which can be addressed by using a parameter known as max-depth to indicate how deep the splitting should be (Microsoft, 2021). LightGBM algorithm was proposed by Su et al. (2018) and has been applied in different studies such as Abbadi et al. (2020) and Fonseca et al. (2017). Abbadi et al. (2020) used LightGBM in IoT malware detection. LightGBM uses Gradient-based One-Side Sampling (GOSS) and the Exclusive Feature bundling (EFB) (Sharma, 2018) to minimize the complexity of histogram building (O(data*feature)). This is achieved by using *GOSS* and *EFB* to reduce the sampled data and feature size. Hence, the complexity becomes (O(data2 * bundles)) where data2 < data and bundles << feature (Sharma, 2018). LightGBM algorithm works on a supervised training set to compute an approximate function that minimizes the value of a specific loss function $L(y, f(x))$ as expressed:

$$\hat{f} = minEy, xL(y, f(x)) \tag{1}$$

where x represents a set of random input variable and y represents a random output or response variable. LightGBM computes an approximation of the final model by combining multiple T regression trees $\sum_{t=1}^{T} f_t(X)$ which is expressed as

$$f_T(x) = \sum_{t=1}^{r} f_t(x) \tag{2}$$

The regression trees could be expressed as $w_{q(x),}q \in \{1,2, \dots, j\}$ where j denotes the number of leaves, q represents the decision rules of the tree and w is a vector that denotes the sample weight of leaf nodes. Hence, LightGBM would be trained in an additive form at step t as follows:

$$\Gamma_t = \sum_{i=1}^{n} L(y_i, F_{t-1}\ (x_{i,}) + f_{t,}\ (x_{i,})) \tag{3}$$

In LightGBM, the objective function is rapidly approximated using Newton's method. After removing the constant term in the last equation for simplicity, the formulation can be represented as

$$\Gamma_t \cong \sum_{i=1}^{n} (g_i f_t(x_{i,}) + \frac{1}{2} h_i f_t^2\ (x_{i,})) \tag{4}$$

where g_i and h_i denote the first and second-order gradient statics of the loss function.

Let I_j denote the sample set of leaf j, (4) could be expressed as

$$\Gamma_t = \sum_{j=1}^{j} ((\sum_{i \in I_j} g_i) w_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i + \Lambda\right) w_j^2\Big)) \tag{5}$$

For a certain tree structure$q(x)$, the optimal leaf weight scores of each leaf node w_j^* and the extreme value of Γ_k can be expressed as:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \Lambda} \tag{6}$$

$$\Gamma_T^* = -\frac{1}{2} \sum_{j=1}^{j} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \Lambda} \tag{7}$$

where

Γ_T^* is the scoring function that measures the quality of the tree structure q. Finally, the objective function after adding the split is:

$$G = \frac{1}{2}\left(\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \Lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \Lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \Lambda}\right) \tag{8}$$

where $I_{L,}$ and I_R are the sample sets of the left and right branches respectively. Unlike traditional GBDT based techniques such as XGboost and GBDT which grow trees horizontally, LightGBM adopts a vertical approach to grow the tree, which makes the algorithm effective for handling large datasets and features. LightGBM increases training performance and minimizes memory requirements by using algorithms based on the histogram. The advantages of LightGBM as presented by Khandelwal (2017) are as follows:

i. Higher efficiency: It uses histogram-based algorithm to convert continuous feature values into discrete bins which results increases the speed of training a dataset.
ii. Reduced memory requirements - the replacement of continuous values with discrete bins results in low memory utilization.
iii. Higher accuracy than similar techniques – by using leaf-wise instead of level-wise splitting to compute more complex trees.
iv. Suitable for large datasets – performs well on large datasets and minimizes the training time considerably when compared to XGBOOST.
v. It supports parallel learning.

3.6 Architecture of XceptionCNN

Xception model was developed using an 'extreme' interpretation of Google's Inception model (Chollet, 2017). Its structure is a linear stack of 36 independent convolution layers which use depth-wise splitting method and are linked together by residual connections. The layered stack is responsible for feature extraction on the network. The 36 independent convolution layers are grouped into 14 modules, which are joined by linear residual connections, with the exception of the first and last modules (Chollet, 2017). XceptionCNN diagrammatic representation is shown in figure 4.

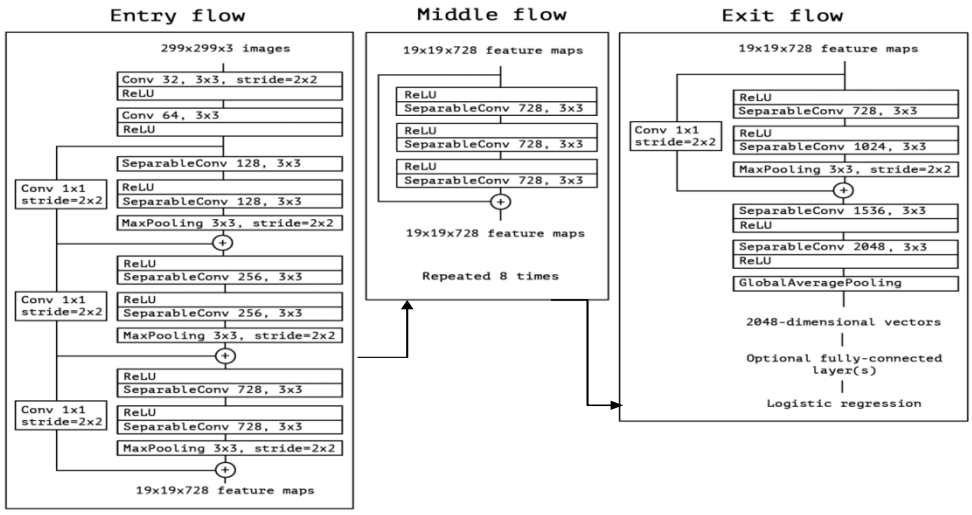

Figure 4: XceptionCNN Architecture (Chollet, 2017)

Figure 4 shows the Xception architecture. It is divided into 3 components (entry flow, middle flow and exit flow), 14 modules and 36 convolutional layers. It uses the layers with a depth of 126 to perform feature extraction. Its input format is a 299x299 RGB image. A global average pooling layer is substituted for the fully-connected layer to reduce the parameter size, while the softmax function is used to predict the output. The flow of data from the entry flow to the middle flow is repeated eight times before it finally passes through the exit flow. The number of convolutional layers in the entry flow, middle flow and exit flow are 8, 8*3= 24 and 4 respectively. The model uses depth-wise separable convolution to reduce the operational cost of the convolution process.

3.7 Design of the XceptionCNN - LightGBM Experiments

To reduce the prediction time and take advantage of the efficiency of XceptionCNN, we combine XceptionCNN with LightGBM. We use the XceptionCNN to extract features automatically; it uses less resource and time as compared to the already available methods. The XceptionCNN is hybridized with the LightGBM model. We directly passed the images that comprise the dataset into XceptionCNN model for automatic feature extraction, thereafter, saved the extracted features as CSV and passed it to the LightGBM model for classification and training.

In order to demonstrate that the XceptionCNN-LightGBM can reduce prediction time and training time, improve the performance and improve the training accuracy, experiments with XceptionCNN-LightGBM were conducted. We compared XceptionCNN-LightGBM models with the generic LightGBM models (preliminary experiment) to

verify that using XceptionCNN-LightGBM can reduce prediction time and training time, improve classification accuracy and training accuracy. We also compared our results with similar studies to show that our model performs better on the Malimg dataset than the previous ones.

In our experiments, we chose LightGBM because of its better accuracy than any other boosting algorithm (Khandelwal, 2017). And also because of, its performance and effectiveness in Robust Intelligent Malware Detection as studied by (Abbadi et al., 2020). Similarly, we chose XceptionCNN because of its efficiency in extracting image features automatically, it is less time consuming and effective (Lo et al., 2019). Figure 5 shows our approach in accomplishing our enhanced LightGBM model.

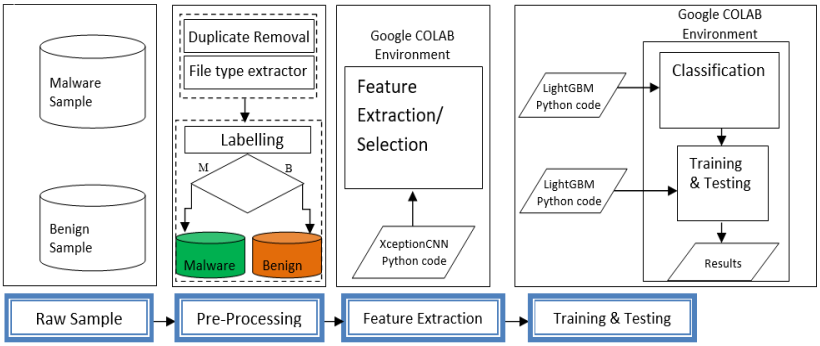

Figure 5: Design of the Proposed XceptionCNN-LightGBM model

3.8 Performance Evaluation

A Confusion matrix N x N (where N is the number of target classes) is used to evaluate the performance of the proposed model. A 2 x 2 matrix is required to perform binary classification. The confusion matrix is used to evaluate the ability of our model to classify the data based on its assigned labels. Each C_{mn} is the data instances which belong to group m (true label) and predicted belonging to group n (predicted label) (Harikrishnan, 2019). C_{00}, C_{01} and C_{10} denote the number of true negatives, false positives, and false negatives respectively.

The basic terminologies used for defining Confusion Matrix include (1) True Positive (TP), which refers to the point at which the predicted positive value matches the real value; (2) True Negative (TN), when the predicted negative value matches the real value; (3) False Positive (FP) - where the real value is negative but the model predicted positive (also referred to as Type 1 error); and (4) False Negative (FN) – a situation where the real value is positive but the model predicted negative (commonly referred to as Type 2 error).

3.8.1 Evaluation Metrics

Evaluation metrics assess the quality of machine learning model in order to obtain necessary feedback and determine its effectiveness and efficiency. We explain the metrics used in evaluating our models below :

i. **Accuracy:** The machine learning model accuracy for a given classification task is given as $\frac{NumberofCorrectPrediction}{TotalNumberofPredictionMade}$

$$\text{Accuracy} = \frac{TN+TP}{TN+FP+TP+FN} \quad (9)$$

Where, *TN, TP, FN*, and *FP* represent True Negative, True Positive, False Negative and False Positive data points respectively.

ii. **True Positive Rate**: True Positive Rate corresponds to the proportion of positive data points that are correctly predicted as positive, with respect to all positive data points.

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP+FN} \quad (10)$$

TP and *FN* are as described in (9).

iii. **False Positive Rate (FPR):** False Positive Rate corresponds to the proportion of negative data points that are mistakenly predicted as positive, with respect to all negative data points.

$$\text{False Positive Rate (FPR)} = \frac{FP}{TN+FP} \tag{11}$$

TN and *FP* are as described in (9).

iv. **Precision:** used to measure the positive patterns that are correctly predicted from the total predicted patterns in a positive class (Hossin, 2015). It is calculated as the ratio of the correct positive results to the number of positive results predicted by the classifier.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{12}$$

TP and *FP* are as described in (9).

v. **Recall -** used to measure the fraction of positive patterns that are correctly classified (Hossin, 2015). It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

$$\text{Recall} = \frac{TP}{TP+FN} \tag{13}$$

TP and *FN* are as described in (9)

vi. **F1 Score:** measures the accuracy and effectiveness of a classifier (Mishra, 2018). The value ranges between 0 and 1. A high F1 score indicates that the model has good performance. It is calculated as the Harmonic Mean of precision and recall. it can be expressed mathematically as:

$$\text{F1 Score} = 2 * \frac{\boldsymbol{Precision * Recall}}{\boldsymbol{Precision + Recall}} \tag{14}$$

vii. **Training Time (TT):** The training time of a model is the total amount of time required for a model to be completely trained. It is the difference between the end time and the start time of training the model.

$$\text{TT} = \text{EndTime(ET)} - \text{StartTime(ST)} \tag{15}$$

3.9 Implementation Environment

We conducted experiments on Google Co-laboratory (COLAB) environment with TPU v3, 32GB HMB. COLAB is a machine learning education and research platform based on Jupyter Notebook (Carneiro et al., 2018). It is pre-configured with necessary machine learning and artificial intelligence libraries, such as TensorFlow, Matplotlib, and Keras. It provides a CPU, GPU and TPU accelerated Python 2 and 3 runtime.

4 Results and Discussion

We performed two experiments involving the Malimg dataset, with binary/multi classification level, using hybrid (XceptionCNN - LightGBM) learning technique. In this section, each experiment will be discussed in detail and results will be presented for the two separate experiments. Each experiment represents the Malimg dataset, binary/multi classification level. The results from our preliminary LightGBM experiments on the Malimg malware dataset are presented in Table 2.

Table 2 presents the results (binary & multi-class) obtained from our preliminary experiment using LightGBM Algorithm**.**

Table 2: Results of LightGBM (Preliminary) Experiments

Classification / Metric	Binary	Multi-Class
Recall	99.80%	96.87%
Precision	99.80%	96.75%
F1_Score	99.80%	96.51%
Training Accuracy	99.80%	96.87%
Training Time	179.51s	2224.77s
Prediction time	0.08s	0.40s

For the binary classification results in Table 1, training time of 179.51s means the model spent a total of 179.51s for training. It obtained a training accuracy of 99.80%, a precision of 99.80% which is the positive patterns correctly predicted from the total predicted patterns in a positive class, and a recall of 99.80%, which means 99.80% fraction of positive patterns, were correctly classified. The Harmonic Mean between this precision and recall which is the F1_score, is 99.80%. The greater the F1 Score, the better the performance of the model. The corresponding multiclass model spent a total of 2224.77s of training time, with training accuracy of 96.87% and a precision of 96.75%, which is the positive patterns correctly predicted from the total predicted patterns in a positive class, with a recall of 96.87%, which implies 96.87% fraction of positive patterns were correctly classified. The Harmonic Mean between precision and recall which is the F1 Score is 96.51%. Similarly, the greater the F1 Score the better the performance of the model. The prediction time of 0.08s and 0.40s for binary and multi classification respectively show the time taken for predictions to occur.

4.1 Experiments on XceptionCNN – LightGBM

In these experiments, we improved the LightGBM model by hybridizing it with XceptionCNN. We installed LightGBM as an independent model on the colab notebook for implementation.

4.1.1 Binary Classification

Binary classification is used to distinguish malware from benign samples. We created the malware class by placing all Malimg families into one malware set. There are 1042 benign samples which are converted to images.

4.1.2 Multi-Classification

We also used the XceptionCNN – LightGBM algorithm to classify malware samples into distinct families. It is a multi-classification problem consisting of 26 classes, The actual Malimg dataset consist of 25 malware families, while the benign set is considered an additional "family" resulting in a total of 26 classes. We present the results obtained from our hybrid experiments in the Table 3.

Table 3: Results of XceptionCNN – LightGBM Experiments

Classification / Metric	Binary	Multi-Class
Recall	99.85%	97.40%
Precision	99.85%	97.29%
F1_Score	99.85%	97.29%
Training Accuracy	99.85%	97.40%
Training Time	29.97s	447.75s
Prediction time	0.08s	0.37s

Table 3 presents the results (binary & multi-class) obtained from our experiments using our hybrid model**.**

For the binary classification results in Table 3, training time of 29.97s means the model spent a total of 29.97s for training. It obtained a training accuracy of 99.85%, a precision of 99.85% which is the positive patterns correctly predicted from the total predicted pattern in a positive class and a recall of 99.85%, which means 99.85% fraction of positive pattern were correctly classified. The Harmonic Mean between this precision and recall, which is the

F1_score is 99.85%. The greater the F1 Score the better the performance of the model. The corresponding multi-class model spent a total of 447.75s training time, with training accuracy of 97.40% and a precision of 97.29% which is the positive patterns correctly predicted from the total predicted patterns in a positive class. A recall of 97.40% was obtained, which means 97.40% fraction of positive patterns were correctly classified. The Harmonic Mean between precision and recall which is the F1 Score is 97.29%. Similarly, the greater the F1 Score the better the performance of the model. The prediction time of 0.08s and 0.37s for binary and multi classification respectively show the time and how fast predictions occur.

4.2 Comparison of Results

The results in Table 3 are comparable to those obtained in our generic LightGBM preliminary experiment (Table 2) and serves to confirm our hybrid model implementation. Table 4 shows the results comparison.

Table 4: Results Comparison

	LightGBM (Preliminary Experiment)		Xception – LightGBM	
Classification / Metric	Binary	Multi-Class	Binary	Multi-Class
Recall	99.80%	96.87%	99.85%	97.40%
Precision	99.80%	96.75%	99.85%	97.29%
F1_Score	99.80%	96.51%	99.85%	97.29%
Training Accuracy	99.80%	96.87%	99.85%	97.40%
Training Time	179.51s	2224.77s	29.97s	447.75s
Prediction time	0.08s	0.40s	0.08s	0.37s

Table 4 presents the results obtained from the various training techniques used. This study shows that it is effective to combine pre-trained XceptionCNN model with LightGBM algorithm to improve detection and classification of windows malware. It leverages on the strengths and benefits of XceptionCNN (Shaheed & Zhang, 2022) and the LightGBM algorithm (Abbadi et al., 2020). Comparing the experimental results obtained in this study with the preliminary experiments, Table 4 shows that combining the pre-trained XceptionCNN model with LightGBM obtains better classification accuracy than applying the generic LightGBM algorithm. Results clearly indicate that extracting image features using XceptionCNN and performing classification using LightGBM provides the best performance for malware detection. In order to evaluate our model, we chose Accuracy, Precision, F1-score, Recall, Training time and Detection time as evaluation criteria. From the results of these experiments in Table 4, we can see that our model achieves a good performance result compared to the preliminary experiments based on the generic LightGBM algorithm. Although the detection time of the binary class looks the same with the LightGBM, figure 7 shows that our hybrid approach outperforms it in terms of detection accuracy of 100%. This is due to the use of pre-trained XceptionCNN model. XceptionCNN extracted fewer misleading features than the generic LightGBM. Hence, fewer misleading data improves modeling accuracy.

4.3 Confusion Matrix

We also present the confusion matrix for our experiments in Figure 6 and Figure 7, to show the improved classification accuracy of our XceptionCNN – LightGBM algorithm.

Figure 6 shows the classification accuracy of the generic LightGBM model. It shows a total data point of 2,077, which corresponds to the 20% (testing data) of the total dataset. The generic LightGBM model performed reasonably well in this malware classification, correctly identifying 99% of the malware samples (True Positive Rate of 99%).

The classification accuracy of the hybrid model is presented in Figure 7. It shows a total of 2,077 data points, which correspond to the 20% (testing data) of the total dataset. Our hybrid model shows an improved classification accuracy of 100% True Positive Rate. This means our model correctly identifies 100% samples that are truly malware.

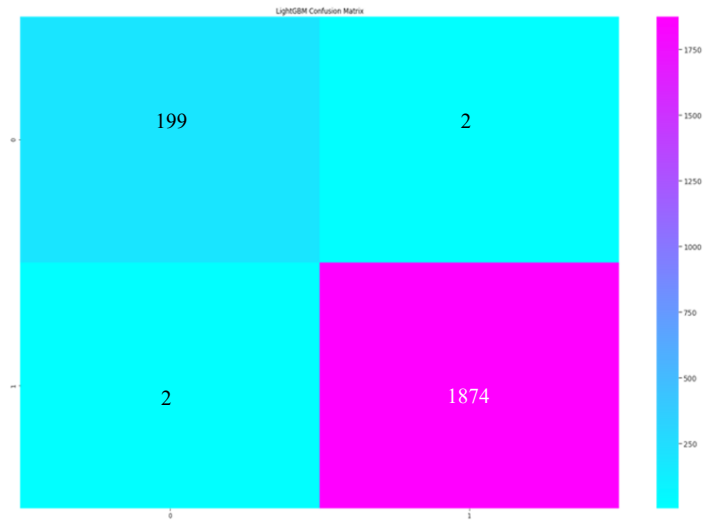

Figure 6: Generic LightGBM Confusion matrix

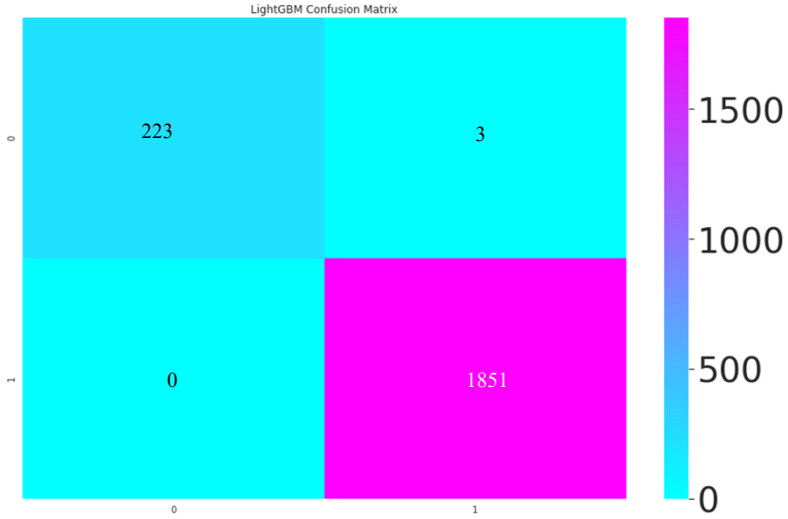

Figure 7: XceptionCNN –LightGBM Confusion Matrix

4.4 Comparison with Related Works

In Table 5, we compare our method with similar studies using the Malimg dataset for Windows Malware detection. Our results maintain higher accuracy than all the approaches in the related work. Our hybrid approach further maintains 100% TPR (see Figure 7).

Table 5: Comparison with Related Works.

Training Techniques	Classification	Training Accuracy
LightGBM (Preliminary Experiment)	Binary	99.80%
	Multi-Class	96.87%
XceptionCNN -LightGBM (Ours)	Binary	99.85%
	Multi-Class	97.40%
M-CNN (Kalash et al., 2018)	Binary	98.25%
DL (Bhodia et al., 2019)	Binary	98.39%
XceptionCNN (Lo et al., 2019)	Binary	99.37%
	Multi-Class	94.80%

The results as presented in Table 5 show that the XceptionCNN – LightGBM model is more effective and robust than previous solutions.

The XceptionCNN – LightGBM model accepts image data as input. The Malimg dataset is available publicly as benchmark Windows malware image dataset used in many studies for image based malware classification. Many machine learning and deep learning algorithms have been presented to develop models for effective malware detection. In this study, we combined XceptionCNN and LightGBM algorithms for binary and multi classification approach for effective and efficient malware detection. The proposed approach was compared with current methods in the literature. Table 5 summarizes the comparison of our performance results with current methods in the literature in terms of Training accuracy. Our model outperforms the state–of–the–art approaches with the best training accuracy. These significant improvements are attributed to the usage of the pre-trained XceptionCNN model and the LightGBM algorithm which aided in producing an excellent outcome. This is because XceptionCNN which is a pre-trained model for image based feature extraction, extracts the best image features needed for training the model by the LightGBM algorithm. The XceptionCNN model extracted adequate and less redundant image features from the dataset which were further trained by LightGBM. Less redundant data means fewer tendencies to make decision based on noise, reduced overfitting and improved robustness. Similarly, Quality training using the LightGBM algorithm produces a highly reliable model, resulting in fast and accurate classification.

5 Conclusion

This study proposed XceptionCNN – LightGBM model for Windows malware detection. The proposed hybrid technique addresses the limitations of the generic LightGBM algorithm in terms of classification accuracy, prediction time, training accuracy and training time. The model was tested on 9,339 malware samples across 25 malware families and 1,042 benign samples. Preliminary experiments based on the generic LightGBM algorithm show a classification accuracy of 99% TPR with prediction time of 0.08s and 0.40s for binary and multi classification respectively. Experimental results show that the hybrid technique provides improved classification accuracy of 100% and reduced prediction time of 0.08s and 0.37s for binary and multi classification respectively. The training accuracy improved by 0.5% and achieved a reduced training time of 29.97s from 179.51s for binary classification and 447.75s from 2224.77s for multi classification. The practical implication of this study is that the hybrid approach provides accurate and reliable detection of malicious software that attack computer systems and compromise the confidentiality, integrity and availability of information stored in them. The reduction in detection time provides early detection of a malware before it causes significant damage to files stored in computer systems. This minimizes the losses an organization will suffer in case of malware attack. The reduction in the training time enables the model to converge quickly and train a large amount of data in a relatively short period of time. A future study will consider using two or more malware datasets to conduct the experiments. This will further enhance the validity and reliability of the proposed model.

References

Abbadi, M. A., Al-Bustanji, A. M., & Al-kasassbeh, M. (2020, April 30). Robust Intelligent Malware Detection using LightGBM Algorithm. *International Journal of Innovative Technology and Exploring Engineering*, *9*(6), 1253–1260. https://doi.org/10.35940/ijitee.f4043.049620

Abusitta, A., Li, M. Q., & Fung, B. C. (2021). Malware Classification and Composition Analysis: A Survey of Recent Developments. *Journal of Information Security and Applications*, *59*, 102828. https://doi.org/10.1016/j.jisa.2021.102828

Bazrafshan, Z., Hashemi, H., Fard, S. M. H., & Hamzeh, A. (2013). A Survey on Heuristic Malware Detection Techniques. *The 5th Conference on Information and Knowledge Technology (*pp. 113-120*).* https://doi.org/10.1109/ikt.2013.6620049

Bensaoud, A., & Kalita, J. (2022). Deep Multi-task Learning for Malware Image Classification. *Journal of Information Security and Applications*, *64*, 103057. https://doi.org/10.1016/j.jisa.2021.103057

Bhodia, N., Prajapati, P., Di Troia, F., & Stamp, M. (2019). Transfer Learning for Image-based Malware Classification. *Proceedings of the 5th International Conference on Information Systems Security and Privacy* (pp 719-726). https://doi.org/10.5220/0007701407190726

Carneiro, T., Nobrega, R. V., Nepomuceno, T., Bian, G. B., De Albuquerque, V. H. C., & Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, *6*, 61677–61685. https://doi.org/10.1109/access.2018.2874767

Chang, J., Venkatasubramanian, K. K., West, A. G., & Lee, I. (2013). Analyzing and Defending against Web-based Malware. *ACM Computing Surveys*, *45*(4), 1–35. https://doi.org/10.1145/2501654.2501663

Chen, J., Guo, S., Ma, X., Li, H., Guo, J., Chen, M., & Pan, Z. (2020). SLAM: A Malware Detection Method Based on Sliding Local Attention Mechanism. *Security and Communication Networks*, 1–11. https://doi.org/10.1155/2020/6724513

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* (pp.1251-1258). https://doi.org/10.1109/cvpr.2017.195

Damodaran, A., Troia, F. D., Visaggio, C. A., Austin, T. H., & Stamp, M. (2015). A Comparison of Static, Dynamic, and Hybrid Analysis for Malware Detection. *Journal of Computer Virology and Hacking Techniques*, *13*(1), 1–12. https://doi.org/10.1007/s11416-015-0261-z

Fang, Z., Wang, J., Geng, J., & Kan, X. (2019). Feature Selection for Malware Detection Based on Reinforcement Learning. *IEEE Access*, *7*, 176177–176187. https://doi.org/10.1109/access.2019.2957429

Fonseca, E., Gong, R., Bogdanov, D., Slizovskaia, O., Gomez, E., & Serra, X. (2017). Acoustic Scene Classification by Ensembling Gradient Boosting Machine and Convolutional Neural Networks. In Virtanen,

T., Mesaros, A., Heittola, T., Diment, A., Vincent, E., Benetos, E., Martinez B. (Eds). Detection and Classification of Acoustic Scenes and Events 2017 Workshop: Tampere University of Technology (pp.37-41). http://hdl.handle.net/10230/33454

Gibert, D., Mateu, C., & Planes, J. (2020). The rise of Machine Learning for Detection and Classification of Malware: Research developments, Trends and Challenges. *Journal of Network and Computer Applications*, *153*, 102526. https://doi.org/10.1016/j.jnca.2019.102526

Harikrishnan, B. (2019, December 10). *Confusion Matrix, Accuracy, Precision, Recall, F1 Score Binary Classification Metric*. National Institute of Advanced Studies, Bengaluru, India. https://medium.com/@harikrishnannb

Hossin, M., & Sulaiman, N. (2015). A Review on Evaluation Metrics for Data Classification. *International Journal of Data Mining & Knowledge Management Process*, *5(2),* 01-11. https://dio.org/10.5121/ijdkp.2015.5201

Huang, K. (2020). An Optimized LightGBM Model for Fraud Detection. *Journal of Physics: Conference Series*, *1651*(1), 012111. https://doi.org/10.1088/1742-6596/1651/1/012111

Hussain, S. J., Ahmed, U., Liaquat, H., Mir, S., Jhanjhi, N., & Humayun, M. (2019). IMIAD: Intelligent Malware Identification for Android Platform. *2019 International Conference on Computer and Information Sciences (ICCIS)*. (pp. 1- 6). https://doi.org/10.1109/iccisci.2019.8716471

Ju, Y., Sun, G., Chen, Q., Zhang, M., Zhu, H., & Rehman, M. U. (2019). A Model Combining Convolutional Neural Network and LightGBM Algorithm for Ultra-Short-Term Wind Power Forecasting. *IEEE Access*, *7*, 28309–28318. https://doi.org/10.1109/access.2019.2901920

Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D. B., Wang, Y., & Iqbal, F. (2018). Malware Classification with Deep Convolutional Neural Networks. *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. https://doi.org/10.1109/ntms.2018.8328749

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *31st International Conference On Neural Information Processing Systems*, (pp. 3149–3157). https://dl.acm.org/doi/10.5555/3294996.3295074

Khandelwal, P. (2017, June 12). *Which Algorithm takes the crown: LightGBM vs XGBOOST?* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-

Kumar, A. (2017). *A Frame work for Malware Detection with Static Features using Machine Learning Algorithms*. [Doctoral thesis] Soongsil University. https://doi.org/10.13140/RG.2.2.35593.90723

Landage, J., & Wankhade, P. (2013). Malware and Malware Detection Techniques: A Survey. *International Journal of Engineering Research & Technology*, *2(12),* 61 - 68. https://doi.org/ 10.17577/IJERTV2IS120163

Lo, W. W., Yang, X., & Wang, Y. (2019). An Xception Convolutional Neural Network for Malware Classification with Transfer Learning. *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (pp.1-5). https://doi.org/10.1109/ntms.2019.8763852

Machado, M. R., Karray, S., & Sousa, I. T. (2019). LightGBM: an Effective Decision Tree Gradient Boosting Method to Predict Customer Loyalty in the Finance Industry. *2019 14th International Conference on Computer Science & Amp; Education (ICCSE)* (pp. 1111-1116). https://doi.org/10.1109/iccse.2019.8845529

Malith, O. (n.d.). *A Simple Utility to Convert EXE Files to PNG Images and Vice Versa*. Github. Retrieved from http://github.com/OsandaMalith/Exe2Image

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. (2018). N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE Pervasive Computing, 17(3), 12–22. https://doi.org/10.1109/mprv.2018.03367731

Microsoft Cooperation. (2021). Read the Docs, *LightGBM Release 3.2.1.99*. Github. Retrieved from https://lightgbm.readthedocs.io/

Minastireanu, E. A., & Mesnita, G. (2019). LightGBM Machine Learning Algorithm to Online Click Fraud Detection. *Journal of Information Assurance &Amp; Cybersecurity*, *12,* 1–12. https://doi.org/10.5171/2019.263928

Mishra, A. (2018, February 24). *Metrics to Evaluate your Machine Learning Algorithm.* Towards Data Science. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. (2011). Malware Images: Visualization and Automatic Classification. *8th* International *Symposium on Visualization for Cyber Security* 2011 (pp.1–7). https://doi.org/10.1145/2016904.2016908

Nawaz, A. (2021). Feature Engineering based on Hybrid Features for Malware Detection over Android Framework. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(10), 2856–2864. https://doi.org/10.17762/turcomat.v12i10.4931

Pan, Q., Tang, W., & Yao, S. (2020). The Application of LightGBM in Microsoft Malware Detection. *Journal of Physics: Conference Series*, *1684*(1), 012041. https://doi.org/10.1088/1742-6596/1684/1/012041

Pant, D., & Bista, R. (2021b). Image-based Malware Classification using Deep Convolutional Neural Network and Transfer Learning. *2021 3rd International Conference on Advanced Information Science and System (AISS 2021)*. https://doi.org/10.1145/3503047.3503081

Şahin, D. Z., Kural, O. E., Akleylek, S., & Kılıç, E. (2021). A Novel Permission-based Android Malware Detection System using Feature Selection based on Linear Regression. *Neural Computing and Applications*, 33, 1 – 16. https://doi.org/10.1007/s00521-021-05875-1

Shaheed, K., Mao, A., Qureshi, I., Kumar, M., Hussain, S., Ullah, I., & Zhang, X. (2022). DS-CNN: A pre-trained Xception Model based on Depth-Wise Separable Convolutional Neural Network for Finger Vein Recognition. *Expert Systems With Applications*, *191*, 116288. https://doi.org/10.1016/j.eswa.2021.116288

Sharma, A. (2018, October 15). *Understanding GOSS and EFB: The core Pillars of LightGBM.* Towards Data Science. https://towardsdatascience.com/what-makes-lightgbm-lightning-fast-a27cf0d9785e

Singh, J., & Singh, J. (2021). A Survey on Machine Learning-based Malware Detection in Executable Files. *Journal of Systems Architecture*, *112*, 101861. https://doi.org/10.1016/j.sysarc.2020.101861

Su, J., Vargas, V. D., Prasad, S., Daniele, S., Feng, Y., & Sakurai, K. (2018). Lightweight Classification of IoT Malware Based on Image Recognition. *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* (pp. 664 - 669). https://doi.org/10.1109/compsac.2018.10315

Sun, X., Liu, M., & Sima, Z. (2020). A Novel Cryptocurrency Price Trend Forecasting Model Based on LightGBM. *Finance Research Letters*, *32*, 101084. https://doi.org/10.1016/j.frl.2018.12.032

Venkat, T., Rao, N., Unnisa, A., & Sreni, K. (2020). Medicine Recommendation System based on Patient Reviews. *International journal of Scientific & Technology research, 9(2)*, 3308 - 3312.

Wang, J. (2018). *Detection and Analysis of Web-based Malware and Vulnerability* [Doctoral thesis]. Nanyang Technological University, Singapore. https://doi.org/10.32657/10220/47659

Wong, M. Y., Landen, M., Antonakakis, M., Blough, M. D., Redmiles, M. E., & Ahamad, M. (2021). An inside look into the practice of Malware Analysis**.** *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (pp. 3053–3069). ACM SIGSAC. https://doi.org/10.1145/3460120.3484759