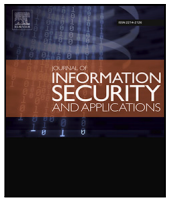
Journal of Information Security and Applications 72 (2023) 103402

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa



A multi-view feature fusion approach for effective malware classification using Deep Learning

Rajasekhar Chaganti [a,*], Vinayakumar Ravi [b], Tuan D. Pham [b]

[a] Department of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249, USA
[b] Center for Artificial Intelligence, Prince Mohammad Bin Fahd University, Khobar, Saudi Arabia

ARTICLE INFO

Keywords:
Cybersecurity
Cybercrime
Malware analysis
Portable Executable
Multi-view
Feature fusion
Machine learning
Deep Learning
Convolution Neural Network

ABSTRACT

The number of malware infected machines from all over the world has been growing day by day. New malware variants appear in the wild to evade the malware detection and classification systems and may infect with ransomware or crypto miners for adversary financial gain. A recent colonial pipeline ransomware attack is an example of these attacks that impacted daily human activities, and the victim had to pay ransom to restore their operations. Windows-based systems are the most adopted systems across different industries for running applications. They are prone to get targeted by installing the malware. In this paper, we propose a Deep Learning (DL)-based Convolutional Neural Network (CNN) model to perform the malware classification on Portable Executable (PE) binary files using the fusion feature set approach. We present an extensive performance evaluation of various DL model architecture and Machine Learning (ML) classifier i.e. Support Vector Machine (SVM), on multi-aspect feature sets covering the static, dynamic, and image features to select the proposed CNN model. We further leverage the CNN-based architecture for effective classification of the malware using different combinations of feature sets and compare the results with the best-performed individual feature set. Our performance evaluation of the proposed model shows that the model classifies the malware or benign files with an accuracy of 97% when using fusion feature sets. The proposed model is robust and generalizable and showed similar performances on completely unseen two malware datasets. In addition, the embedding features of the CNN model are visualized, and various visualization methods are employed to understand the characteristics of the datasets. Further, large-scale learning and stacked classifiers were employed after the penultimate layer to enhance the CNN classification performance.

1. Introduction

The advancement in Information and Communication Technology (ICT) has made everyone connected virtually and rely on computer systems. On the other side, an adversary may take advantage of the machine's weaknesses in the device and compromise the computer machines with malware . Novel malware creation and hiding approaches are emerging to constantly evade malware detection and successfully install the malware on victim machines. Recent trends in malware threats show that more than 8 billion malware attacks are performed per year [1]; 560,000 instances of new malware are being created and detected every day, and 1 in every four machines is likely infected with malware in US [2]. The working from the home situation due to COVID-19 may even pose more risk to malware infection, as the devices are exposed to the public internet with few security controls in a home network environment.

The literature's well-known malware detection techniques include signature-based, heuristic-based, and sandboxing . Signature-based techniques are able to detect the known malware in the wild with defined rules. But, it has a disadvantage of difficulty in identifying new malware variants and requires a constant update of signature rules. Heuristic-based detection systems monitor the anomalies in the system, network, and user behavior to detect the malware. It may detect the new malware attacks and malware variants possessing at least some behavioral characteristics. However, these systems will generate more false-positive alerts and require manpower to tune the alerts. Additionally, it may not be able to detect Advanced Persistent Threat (APT) and well-crafted malware. Sandboxing technique is applied to monitor the real-time behavior of malware for detection. Although we can determine the impact of the malware on a machine using sandboxing, An adversary may use sophisticated techniques to not execute the malicious executable files in a virtualized sandboxing environment and avoid the detection. In recent years, next-generation anti-malware detection systems leveraging ML and DL techniques to

* Corresponding author.
E-mail addresses: Raj.chaganti2@gmail.com (R. Chaganti), vravi@pmu.edu.sa (V. Ravi), tpham@pmu.edu.sa (T.D. Pham).

https://doi.org/10.1016/j.jisa.2022.103402

Available online 14 December 2022

identify advanced malware attacks. ML solutions require feature selection and large scale data. In the context of malware detection, some features can be extracted from malware binaries including PE header, PE section details, byte histogram, opcode n-gram, API call sequences but not limited [3]. There are mainly two ways to extract the malware features, those are using static and dynamic malware analysis. Static features extract meaningful information with regard to the composition details of the file. PE section, PE import functions, PE header, byte, and Opcode histograms are commonly used in static malware analysis [4]. But, these features may overlook essential information related to sophisticated malware techniques like Obfuscation, Metamorphism, Polymorphism, and Oligomorphic code used to avoid detection. Dynamic malware analysis may capture behavioral features of the executable file for malware detection and classification. The most commonly used feature set in dynamic malware analysis is API call sequence [5] because it captures the interaction of the binary with various system resources and also sees the intention of the malware creation. Researchers also utilized the combination of the static and dynamic features called hybrid analysis to accurately detect threats and improve the performance [6–8]. In recent trends, DL models gained popularity to use in cybersecurity and particularly, in malware detection applications [9–11] because they have shown good performance, feature engineering steps can be avoided, and domain knowledge not needed for data analysts. DL models could take the whole binary raw bytes as input features to obtain the best results [12]. We have also witnessed that DL models may achieve better performance than ML techniques for malware classification [13,14]. In order to best utilize DL capabilities seen in computer vision applications, image-based feature extraction is widely used as a third way of retrieving the features from executable files. In image-based feature extraction, the malware executable files can be transformed as a color or grayscale image; represent the color or grayscale image features in input form, and then apply DL models like CNN to achieve performance results [10,15,16]. These image-based malware detection and classification methods are platform independent and may detect the packed malware. It is also evident that image-based feature is used in conjunction with static features [17,18] to improve the performance.

Most of the existing literature work shows that the malware detection using a single feature set either API calls [19,20] or other PE features extensively, and few works using the combination of the two feature sets like hybrid malware analysis. However, there is no prior art detailed investigation and analysis of the malware detection of the various feature sets and the fusion feature sets utilizing multiple aspects of a malware file. In this work, we leverage the static features PE section details, PE import functions, dynamic feature API call sequences, and binary image features as feature sets to perform malware detection and classification using various DL approaches and present the best performed as the proposed model for our approach after extensive performance analysis and validation. The major contributions of the proposed work are as follows

- Propose multi-view feature fusion-based feature selection approach for effective and robust malware detection and classification.
- Present detailed analysis and investigation of various DL model architectures and ML techniques for malware classification.
- Performance evaluation of all the models on static, dynamic, and image feature sets, and the combinations of the fusion feature sets are discussed.
- Describe the layer wise performance of the proposed DL CNN model and feature visualization using t-distributed stochastic neighbor embedding (t-SNE) dimensional reduction.
- Performance evaluation of penultimate layer features of the proposed model is shown with large-scale learning and meta-classifiers.
- Various experiments are included on two unseen malware datasets to show that the proposed method for malware classification is robust and generalizable across unseen malware data samples.

2. Literature survey

Malware detection has been a vivid area of research and various approaches were proposed for malware detection. The detailed study and analysis on malware detection, particularly, windows executable malware detection are described in [3,21,22]. Gibert et al. in [3] performed a comprehensive survey of the malware detection and classification using ML techniques and also discussed recent trends leveraging DL approaches to defend against malware attacks. The survey is categorized based on the PE feature types extracted from static or dynamic analysis and various ML/DL techniques used to detect the malware by utilizing various feature types. Abusitta et al. in [21] designed a framework for analyzing the existing malware classification and composition analysis and also presented a review of the articles describing the features and algorithms used in those articles. In [22], the authors classified malware detection approaches into signature, heuristics, behavior, model checking, DL, Internet of Things (IoT)-based, cloud computing-based, and mobile-based. Overall, all these survey papers emphasize that malware detection using various techniques mentioned in the prior art still has challenges to accurately detect the malware and in particular production environments, as the sophistication of malware creation by adversary ever-changing and adversary always find new ways to evade the existing detection models.

2.1. Static features

Malware feature data collection is one of the primary and important tasks . in the process of applying ML or DL models for malware detection and classification. The quality of the feature data helps to best train the models and obtain better performance by distinguishing the benign and malicious executable files. There are a number of tools available for free to perform PE static file analysis and extract the features like PE imported functions, PE headers, PE section details, byte N-grams, opcode N-grams, and strings [23]. Raff et al. in [12] presented a neural network solution to feed the whole PE executable raw bytes as an input sequence. The paper focused on addressing the challenges that arise when the models learn the long sequence of raw bytes with over two million time steps to produce the malware classification output. Their proposed architecture “MalConv” used a CNN to apply on long sequences. The results obtained showed that neural network architectures have the potential to be used for accurate malware classification, even though it has challenges to handle long sequences of data like executable static raw data. Vinayakumar et al. in [13] proposed “DeepMalNet” 10 hidden layer Deep Neural Network (DNN) architecture to classify the given file is malware or benign using the PE static file information, header information, imported and exported functions, section information and format agnostic features byte histogram, byte entropy histogram, string information. The paper showed that DNN with considerable hidden layers performed well compared to the classical ML algorithms.

In [24], the authors described a low resource and highly accurate DNN-based malware classification method. The static features byte histogram, entropy histogram, PE metadata features, and PE import features are extracted as 256 feature vectors each from the executable file. Then, the four 256 feature vectors combined to form 1,024 feature vectors as an input set. The proposed two hidden layer DNN model achieved low false positive rates and can be scalable to deploy in a cloud analytics platform. Azeez et al. [25] presented an ensemble learning based malware detection using PE static features. The first stage classifier consists of a stacked ensemble of fully connected CNN. The final stage layer is tested with multiple machine learning algorithms for choosing the best performance model. The obtained results showed that an ensemble of seven neural networks in the first stage and an ExtraTree classifier in the final stage achieved the best results for malware. Overall, these papers conclude that the carefully selected static features along with applying DL models may result in fruitful

Table 1
Prior art work comparison of feature selection and the proposed methods for malware classification.

Authors	PE Section	PE Import	PE API	PE Image	Feature info	Feature fusion	Method
Raff et al. [12]	Yes	Yes	No	No	Raw Bytes	Partial	CNN
Saxe et al. [24]	Yes	Yes	No	No	Byte histogram, Entropy histogram, PE metadata features, PE import features	Partial	DNN
Vinayakumar et al. [13]	Yes	Yes	No	No	Header info, string, Byte and Opcode histogram, PE section	Partial	DNN
Li et al. [26]	No	No	Yes	No	API call sequence	No	LSTM,GRU
Zhnag et al. [27]	No	No	Yes	No	API call sequence	No	Gate CNN, Bi-LSTM
Choi et al. [8]	No	No	Yes	No	Opcode sequences	No	CNN,LSTM
Vasan et al. [15]	No	No	No	Yes	Malware images	No	CNN
Jain et al. [10]	No	No	No	Yes	Malware images	No	CNN, EMM
Sun et al. [11]	No	No	No	Yes	Malware images	No	RNN, CNN
Huang et al. [6]	No	No	Yes	No	Byte frequency	Partial	CNN
Proposed	Yes	Yes	Yes	Yes	PE section, PE import, PE API, PE image	Yes	CNN

malware classification performance results. But, the static features standalone may not detect the sophisticated malware. For example, the malware contains encrypted code, and code that can be decrypted during execution is hard to detect using PE static features.

2.2. Dynamic features

The behavioral characteristics of an executable file can be useful for the accurate detection of malware executables, in particular well-crafted when the well crafted malware functionality is hidden in files. So, it is crucial to capture the behavioral features while applying ML or DL models for malware classification. But, the dynamic analysis of an executable requires to be run in an isolated or virtual environment to extract behavioral features. Li et al. in [26] presented an API call sequence-based malware classification using Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models. The API call sequences for the malware samples are generated using a cuckoo sandbox environment. The performance results presented in the paper show that precision and recall for the malware families classification is 75%, which is not impressive.

In [9], the authors proposed a DL model using dynamic features as part of the intelligent malware detection approach. The dynamic features packets sent/received, number of processes running, bytes sent/received CPU and memory usage [28] were considered to perform the malware classification using DNN and CNN. The results concluded that the DL models performed better than ML models and CNN can more accurately classify the malware than DNN. Zhang et al. In [27] implemented a feature engineering method to extract the meaningful information from the API call sequences. Then applied two GatedCNNs and BiLSTM models for malware classification. Hashing tricks were used on API name categories, and arguments to create dataset features. The model was evaluated using API call sequences as data features and compared with the key results. Our survey shows that API call sequence [8,29,30] is widely used as a key dynamic feature for identifying the malware using ML or DL solutions.

2.3. Image representation

In general, neural networks have been extensively used for Computer Vision and Natural Language Processing (NLP) applications. Researchers also explore effective malware detection solutions by converting the executable files into an image and then applying neural network models. Sun et al. in [11] presented a feature image generation approach and applied CNN to classify the feature images. The static code is fed to a recurrent neural network (RNN) to generate predictive code and then performs a fusion of the predictive code from RNN and static code to generate feature images using minhash. These feature images are trained using CNN to predict the malware feature images. The authors reported that the described model achieved 99.5% when training to validation sample size proportion of 3:1. In [10], The authors applied Extreme Learning Machine (ELM) and CNN techniques to classify the malware using a grayscale represented image malware dataset. The reported results mention that ELM achieved comparable accuracies with CNN, despite ELM randomly initializing the weights between input and hidden layers.

Vasan et al. [15] proposed an ensemble CNN architecture for image-based malware classification. The authors employed VGG16 and ResNet-50 network architecture and also applied the softmax as well as one-vs-all multiclass SVM's to ensemble the output and predict the malware classification. Overall, These image-based malware classification results show that the DL models in particular CNN architectures are extensively used for malware classification and achieved good results. But, a careful selection of architecture is required to design a less time, low resource, and high performing data model when selecting the image features of malware.

Table 1 presents the state-of-the-art papers focused on using static, dynamic, image features, and hybrid features and their proposed methods along with the comparison of the proposed feature fusion approach for windows malware classification. These prior arts show that most of the works considered single view feature sets either static, dynamic, or image-based to solve the malware classification problems, and none of the existing state-of-the-work approaches have considered all the four feature sets such as PE Section, PE Import, PE API, and PE Image for capturing the multiple features of the malware for classification. The importance of considering all these four feature sets is evaluated and discussed in detail in this work.

The existing feature extraction techniques static, dynamic, and image representation from an executable file can represent certain characteristics of an executable file. It is very likely to evade malware detection if trying to detect advanced persistent threats using a specific feature set.

2.4. Multiview feature fusion approach

Multiview feature based learning approaches received attention for use in cybersecurity applications because they improve the model performance compared to the single view feature learning approaches [31,32]. A single view feature learning may cover a specific view of the data samples and may not effectively learn all the essential data patterns information within the training datasets. In multi view feature learning, each feature dataset view presents a unique semantic perspective of the data sample. The combination of multi view features can learn the different semantic perspectives of the data sample. It will improve the model accuracy. Some works explored the application of multi view based feature learning for malware detection, and classification [31,32]. Table 2 shows the comparison of the state of the art multi view feature based works for malware classification and Advanced persistent threats family attribution [31–35]. The articles [31–33] applied the multi view feature datasets to detect the android malware and malware family classification. The Opcodes, bytecode, header, API calls, and permissions feature datasets are commonly used to capture the multi view of the android malware and perform feature fusion to combine all the different view features. Then, machine learning or deep learning models are applied to detect and classify the android malware. The articles [34,35] applied a multi view feature learning approach to attribute the APT malware with threat groups. The opcodes, bytecode, API call, and header information are used to build the multi view feature dataset and apply machine learning or deep learning for APT threat attribution. However, the multi view feature approach for the

Table 2
Multiview malware detection or threats attribution prior art comparison.

Authors	Number of views	Features	Operating system	Feature fusion	ML/DL technique	Application
Appice et al. [31]	10	Static features	Andriod	Yes	Clustering, Random Forest	Malware detection
Millar et al. [32]	4	Opcodes, permissions, Arbitary API package, Proprietary Android API packages	Andriod	Yes	CNN	Malware Detection
Darabian et al. [33]	2 or 3	OpCodes, ByteCodes, header information, permission, attacker's intent and API call	Windows, IoT and Android	Yes	KNN, RF, Adaboost, DT, MLP, SVM	Malware detection
Haddadpajouh et al. [34]	12	Opcode, Bytecode, SystemCall and Header	Generic	Yes	Clustering and Decision Tree	Cyber threat attribution
Sahoo [35]	11	Opcode, Bytecode, and Header features	Generic	Yes	SVM, Decision Tree, KNN, MLP, and Fair Clustering	Cyber threat attribution
Proposed	4	PE section, PE import, PE API, PE image	Windows	Yes	CNN	Malware Detection

Table 3
1D-CNN and 2D-CNN based malware classification comparison.

Author	1D CNN	2D CNN	Technique	Application	Performance
Sun et al. [11]	–	Yes	Malware image features	Image based malware family classification	92-99.5% accuracy
Huang et al. [6]	–	Yes	Malware static and dynamic features as an image	image based malware binary classification	94.7% accuracy
Miller et al. [32]	Yes	–	Multi view feature fusion	Zero day malware classification	91% accuracy
Zhnag et al. [27]	Yes	–	API features with Gated CNN	Malware binary classification	98.71±0.17 AUC
Azeez et al [25]	Yes	–	CNN as first stage classifier	Malware binary classification	97.7% accuracy
Nisa et al. [7]	–	Yes	CNN based AlexNet and Inception-V3	Malware family classification	99.3% accuracy
Cui et al. [16]	–	Yes	Malware image features	Image based malware family Classification	94.5% accuracy
Proposed	Yes	–	Multi view feature fusion	Malware binary classification	97.7% accuracy

windows executable file malware detection is not been extensively used in the prior art study. Furthermore, the multi view of the malware feature sets, including the static, dynamic, and image feature fusion based study is not seen in state-of-the-art. We address the windows malware detection using multi view feature datasets. The feature fusion and the machine learning or deep learning methods are used to detect the malware accurately.

2.5. 1D CNN vs 2D CNN in CyberSecurity applications

Convolutional Neural networks (CNN) are commonly used for image classification problems. The image is represented in two-dimensional form, and the convolutional layers followed by pooling layers are applied to the images for image classification. In the context of malware classification, the malware file sequence of bytes is represented in the image form. Some prior art works converted malware files as an image and applied CNN to classify the given file is malware or not [6,10,11,15,36]. Table 3 compares the one-dimensional and two-dimensional CNN techniques used to perform the malware binary classification and malware family classification. Some works leverage the pretrained image classification models like Alexnet and InceptionV3 to perform the malware classification [7]. The static and dynamic features are usually represented in the one-dimensional dimensional feature vector. The one-dimensional CNN can be used to capture the essential information from the one-dimensional feature vector. The one-dimensional CNN was applied to extract the features and classify the malware in the prior art [25,27,32]. State of the art indicates that both one-dimensional and two-dimensional CNN's used to classify the malware. The selection of the CNN depends on the feature type and malware file representation dimensions. The image representing malware or benignware file samples is further processed to denote it as a one-dimensional image feature vector in our work. So, we use one-dimensional CNN to process the one-dimensional feature fusion input and classify the malware in our work.

Our work is motivated to some extent from the prior art papers [37–39]. But, we focus on combining the multiple aspects of an executable file for robust malware detection and also performing an extensive evaluation of the fusion feature sets and individual feature sets using various DL models and SVM to unravel the advantages of proposed multi-view feature fusion set approach.

3. Proposed approach

The proposed DL-based multi-view feature fusion malware classification approach to gathering static features PE section details, PE importing API function calls, dynamic features API call sequences, and binary transformed image features from malware samples; fusing all the selected features to incorporate multiple feature characteristics to distinguish the malware executable file from the benign executable; adapt the best performed DL-based model CNN on the individual feature datasets and implement the proposed best performed model CNN to evaluate the performance on the feature fusion dataset as well as perform the detailed comparative analysis along with prominent ML and DL models used in malware detection and classification.

3.1. Proposed deep learning architecture

The optimal algorithm selection is pertinent for the accurate detection of malware executables. In order to estimate the feature engineering based supervised learning model, we have considered the best performed model in the past, SVM. As feature engineering tends to be time-consuming and tedious, DL models were seen to be used extensively in malware categorization, and performance results were achieved as per the expectations in the state-of-the-art. Hence, the well-known DL models used in malware detection and classification with inputting different types of features are selected for our evaluation. Those models are DNN, CNN, and LSTM, and the combination of CNN and LSTM. Furthermore, to select the optimal hyperparameters for DL models, different versions of the models are tested by changing the hidden layers between the input and output layers. As one of our objectives is to show the performance comparison of the single view of feature sets and the combined multi-view of fusion feature set, we have selected the best performed model, which showed optimal performance on the majority of the individual feature sets and also showed comparable performance on non-optimal result feature sets. On the basis of our extensive performance evaluation with a different set of DL models on each individual feature set, we opted suitable CNN model for our feature sets elected from static, dynamic analysis, and binary to image conversion methods methods. Consequently, the CNN model has been used for our feature fusion combination dataset evaluation to perform the malware classification.

The pseudocode for the CNN model used in our feature fusion approach is shown in Algorithm 1. Here, we assume that the four feature sets are already preprocessed to contain the same malware sample features in each feature set and not presented in pseudocode. In each epoch 1 to e, the fusion feature samples 1 to k are passed through the CNN model to classify whether the sample is malware or not.

3.2. Description of the deep learning model for our approach

The optimal CNN architecture obtained after careful evaluation of the performance results is illustrated in Fig. 1 to process the combined fusion features of the datasets. As represented in Fig. 1, the input layer comprises two static feature PE section details, PE imported API functions datasets, Dynamic feature PE API call sequence dataset, and PE binary transformed to image feature dataset taken to cover the

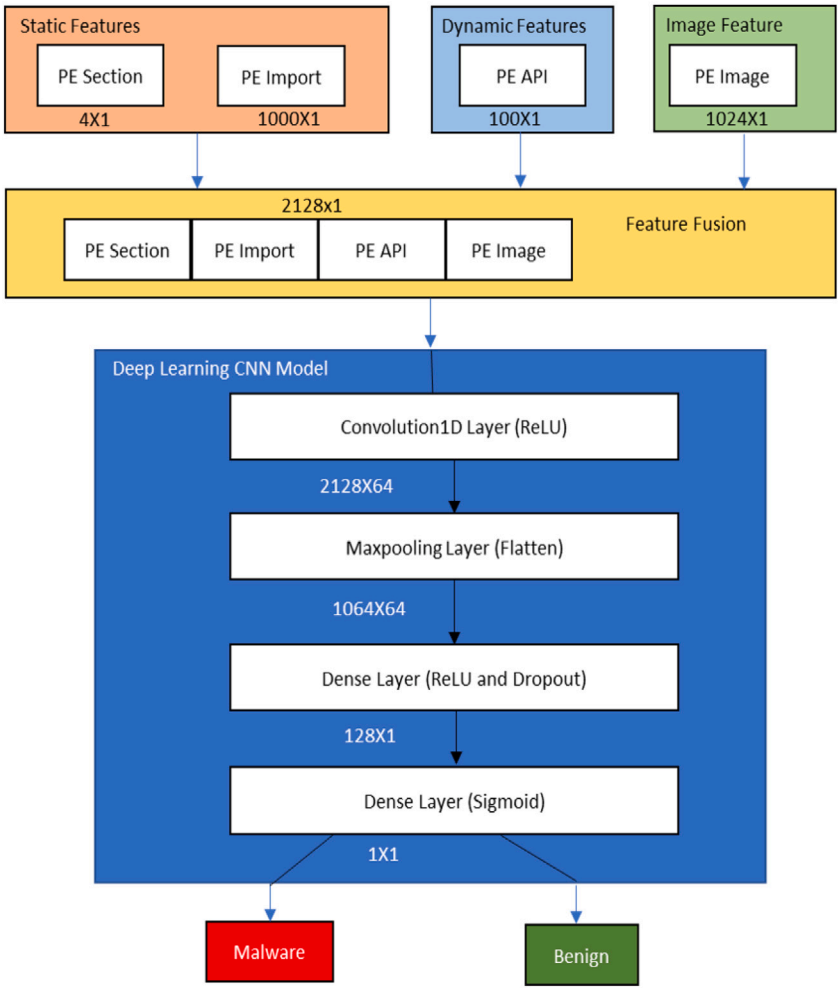


Fig. 1. Proposed CNN based multi-view feature fusion approach.

Algorithm 1: CNN based feature fusion algorithm for malware detection

Input: PE section: $\{s_1, s_2, \ldots, s_m\}$ PE import: $\{i_1, i_2, \ldots, i_n\}$ PE API: $\{a_1, a_2, \ldots, a_p\}$ PE image: $\{im_1, im_2, \ldots, i_q\}$, fusionfeature=$FF_1, \ldots, FF_t$, m+n+p+q=t, kernelfilter=k, datasetsamples=d, epoch:e,denseunits:x, filterlength=l,poolsize=b

Output: Labels $\{Malware, Benign\}$

$FF_1, \ldots, FF_t \leftarrow$ Featurefusion($\{s_1, s_2, \ldots, s_m, i_1, i_2, \ldots, i_n, a_1, a_2, \ldots, a_p, im_1, im_2, \ldots, im_q\}$);

$FS_1, \ldots, FS_t \leftarrow$ Preprocessing($FF_1, \ldots, FF_t$) ;

for $epoch \leftarrow 1$ **to** e **do**

  for $j \leftarrow 1$ **to** d **do**

    $C1_{1j1}, \ldots, C1_{tjk} \leftarrow$ Convolution1D($k, l,'relu', FS_{1j}, \ldots, FS_{tj}$);

    $M2_{1j1}, \ldots, M2_{t_{/2}jk} \leftarrow$ Maxpooling1D(b, $C1_{1j1}, \ldots, C1_{tjk}$);

    $F3_{1j1}, \ldots, F3_{t_{/2}jk} \leftarrow$ Flatten($M2_{1j1}, \ldots, M2_{t_{/2}jk}$);

    $D3_{1j}, \ldots, D3_{xj} \leftarrow$ Dense(x, $F3_{1j1}, \ldots, F3_{t_{/2}jk}$);

    $D5_{1j}, \ldots, D5_{x_{/2}j} \leftarrow$ Dropout($D4_{1j}, \ldots, D4_{xj}$) ;

    O_1 =Sigmoid($D5_1 j, \ldots, D5_{x_{/2}j}$);

  end

end

multi-view of executable files. These input datasets were clubbed together to obtain the fusion feature dataset having 2,128 unique features from each dataset. The order of the feature fusion is also needed to train the CNN model and ensure that the CNN model learns the features when training the datasets. So, a consistent feature fusion order should be used to train and test the models. Our motivation for this work is that the combined features can complement each other's advantage and achieve better performance results, which is described in more detail in the results discussion Section 5. We have fed the fusion feature dataset as an input to the best results achieved CNN model to test our hypothesis. The selection of the CNN model is based on the best performance achieved among all the models evaluated when the four individual feature datasets are applied as an input separately. The input data pass through multiple layers to extract the meaningful information in the CNN model, and the detailed description is explained as follows.

Convolution Layer: The essential component convolution filter in CNN is applied to the input feature fusion dataset. As the final executable binary is represented in a 1-dimensional vector in our datasets, the 1-dimensional convolution layer is chosen in the model. The convolution filters 64 were applied to the input with kernel size 3 to map the features. The activation functions $ReLU$ is opted to add nonlinearity.

Pooling Layer: A pooling map of size two is applied to the assembled feature maps obtained from the convolution layer. The pooling layer helps to reduce the dimensionality and the amount of time to perform computations, and the number of learning parameters . Furthermore, we have applied flatten function to reduce the output data of the pooling layer into 1-dimensional data.

Dense layer: After pooling layer and flatten to 1-dimensional data, it passes through the fully connected layer to perform malware detection. The fully connected layer contains 128 units, including the activation function as $ReLU$. It combines all the features from the input to obtain optimal features. Then, the dropout layer randomly selects some values to 0 based on the chosen value between 0 and 1. We have considered the value 0.5 to drop half of the fully connected layer output data for redundancy. Subsequently, the $sigmoid$ function is applied to obtain the normalized probability distribution values between 0 and 1 to classify whether the feature dataset sample is benign or malware.

4. Datasets

In this section, we describe the details of the datasets considered for our malware classification experiments and performance analysis, and also describe the process used to collect the datasets.

Four different feature datasets were selected for this malware classification study, and all these datasets were generated using PE malware and benign file samples. The malicious applications downloaded from viruatotal.com and benign applications collected from windows7 x86 directories and portableapps.com [40]. These datasets have a naming convention, which starts with "PE" and ends with a feature name extracted from either the image or static or dynamic analysis of the executable. For instance, the dataset "PE API" represents the API calls performed by the PE application when executed in a virtual environment. A detail description of these four datasets is given below. We follow the same convention for representing datasets throughout this paper.

The dataset 1 "PE section" data consists of the PE file section information. These static data features are collected by executing the malware samples in a cuckoo sandbox environment and saving PE sections report [40]. 4 PE section header features are extracted from each binary sample. The collected data contain unique 38,442 PE malware samples and 875 benign samples.

The dataset 2 "PE import" data comprises the top 1,000 imported functions extracted by running the PE in the Cuckoo sandbox environment and generating the static pe_import report [41]. The dataset contains unique 38,442 malicious file samples and 876 benign files.

The dataset 3 "PE API" was generated by running the malicious and benign applications in the cuckoo sandbox virtual environment and saving the first 100 non-repeated consecutive API calls from the parent using the "calls" feature in cuckoo sandbox [42]. This dataset comprises 42,797 malware API call sequences and 1,079 benign API call sequences.

The dataset 4 "PE image" contains PE malware or benign file represent in image form. The nearest neighbor interpolation algorithm is applied to the PE application raw byte stream to convert into the 32 x 32 grayscale [43]. The image is then transformed as a 1,024 bytes vector. This dataset comprises 38,443 malicious file samples and 875 benign file samples.

All these four datasets can be compared or correlated using the column file hash value. Each malware file has a unique hash value. For instance, a malware sample feature datasets PE section, PE import, PE API, and PE images combined using the unique hash value so that the same malware static, dynamic, and image features are combined as the fusion features for the malware. The process is repeated for the benign sample PE files as well. So, each malware or benign file contains 2128 fusion features, which represents the same malware or benign file. The feature fusion dataset also includes a column "Label". The "Label" column is 0 for benign samples and 1 for malware samples. In this way, we make sure the malware or benign sample file associated with four dataset features merged correctly and labeled correctly as malware or benign file.

Then, we randomly selected 1,500 malware samples from a pool of more than 38000 malware samples to balance the malware and benign file proportions. The benign PE executable files were taken from portableapps.com and the 32-bit Windows 7 ultimate directory. The number of publicly available benign PE files is limited. The malware samples were extracted from virtustotal.com. So, the considered dataset is highly imbalanced, with around 38000 malware and 875 benign PE files. We have chosen 1500 malware samples for our experiments because the malware versus benign file proportions should be balanced, align with the computing resources available for our experiments, and should not overfit the model for malware classification. Our selection of 1500 samples ensures that the training model is not overfitting or underfitting and will not have a drastic impact on the test malware classification performance. We also tested our model on the unknown malware samples to validate our training model and evaluate the performance.

Table 4
The four dataset malicious and benign file sample statistics.

Dataset	Raw data		Random selection	
	Malware	Benign	Malware	Benign
PE Section	38,442	875	1,500	875
PE Import	38,442	876	1,500	875
PE API	42,797	1,079	1,500	875
PE Images	38,843	875	1,500	875

The detailed four dataset malware and benign file sample statistics are shown in Table 4.

The final datasets having unique 2,375 file samples each is randomly split into training and test dataset samples with 73% and 27% proportions to perform ML and DL performance evaluation and comparative analysis; the total number of training and testing data samples are described in Table 5. Also, the table contains the total number of malware and benign samples in training and test data for all four datasets.

5. Results and discussion

The experiments were performed using software application stack ML library Scikit-learn[1] and the DL API Keras[2] with TensorFlow[3] as backend in python and the software programs run in virtual machine 64-bit Ubuntu operating system 20.04 LTS with configuration 4 GB RAM memory and Intel Core i5-4210U CPU@1.70 GHz.

5.1. Evaluation metrics

The performance evaluation of the data analytics models is represented in the form of metrics. One of the columns in the datasets is labeled as "benign" or "malware" for malware classification. Table 6 shows the malware classification metrics in the form of a confusion matrix.

True positive (TP) is measured as the correct detection of the malware p when the malware p′ is present. False positive (FP) is determined as the incorrect classification as malware P when no malware n′ is present. False negative (FN) is measured as the incorrect classification as legitimate n when the malware P′ is present. True negative (TN) is determined as the correct classification as the legitimate N when no malware N′ is present.

Our work uses the following evaluation metrics for comparative analysis of ML and DL models.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = TPR = \frac{TP}{TP + FP} \tag{2}$$

[1] https://scikit-learn.org/stable/
[2] https://keras.io/
[3] https://www.tensorflow.org/

Table 5
The train and test dataset malware and benign sample statistics.

Dataset	Train			Test		
	Malware	Benign	Total	Malware	Benign	Total
PE Section	1,100	634	1,734	400	242	642
PE Import	1,100	634	1,734	400	242	642
PE API	1,100	634	1,734	400	242	642
PE Images	1,100	634	1,734	400	242	642
Total	4,400	2,536	6,936	1,600	968	2,568

Table 6
Table of confusion.

		Malware Prediction p	Malware Prediction n	total
Actual Malware	p′	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 - Score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{4}$$

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

$$AUC = \int_0^1 \frac{TP}{TP + FP} d \frac{FP}{FP + TN} \tag{6}$$

Accuracy is defined as the sum of the true positives and true negatives to the sum of the true positive, true negative, false positive, and false negative. Precision is the ratio of the true positive to the sum of true positive and False positive and also called as a true positive ratio. The recall is the ratio of the true positive to the sum of true positive and false negative. The higher the values of precision and recall of a model, the better the model is performed. F1-score is the harmonic mean of precision and recall.

The false positive rate is the ratio of the false positives to the sum of false positives and true negatives. Area Under Curve (AUC) represents the area under the receiver operating characteristic (ROC) curve. It shows the performance comparison using the metrics true positive and false positive ratios under all possible classification thresholds.

Firstly, the individual feature datasets such as "PE section", "PE import", "PE API", and "PE image" performance is evaluated using ML and DL models. These datasets are balanced and preprocessed using the standard scalar function for standardization. We have considered the classical ML model SVM and the DL DNN, CNN, LSTM, and CNN-LSTM for our evaluation of these datasets. The DL models with multiple hidden layers are considered for our tests. The notation "DL model with a number of hidden layers" is followed to distinguish the different models. For instance, DNN1 signifies that the DNN model with 1 hidden layer is used for the experiments. We considered DNN and LSTM models, varying the hidden layers from 1 to 4 for our evaluation. Table 7 shows all the hyperparameters considered for DL models DNN, CNN, LSTM, and CNN-LSTM. All the experiments run for 200 epochs, and we have chosen the number of dense layers in DNN and the LSTM layers in LSTM model-based on the number of hidden layers in the model. The batch size 64 is selected for all DNN models and 32 for all LSTM models. Dropout layer with a fraction of the input dropped varying from 0.01 to 0.1. In addition, $ReLU$ activation function is used in DNN and CNN models. The number of filters 64 and kernel size 3 is chosen for 1-dimensional convolution in the CNN model.

The selection of the hyperparameters in our model evaluation relies on the computational power to run the experiments, the input features size, and the performance of the trained datasets. The number of epochs selected depends on how faster the optimal performance is achieved during the training. We have considered the total number of epochs 200 to run the experiments while ensuring that the computing resources are available to complete the experiments. The standard 'Relu' function is chosen because it gives better convergence and is computationally efficient. The batch size selection depends on the input features dataset size. We have selected the batch size 64 for DNN and 32 for LSTM to train our model with 2128 features. The number of filters, 64, and the kernel size three were selected for CNN evaluation based on the input feature size. The dropout factor is chosen as 0.01 to drop a few redundant feature information in DNN and 0.1 drop factor to ignore the redundant feature information in LSTM. For the CNN model, the dropout was much higher at 0.5 compared to the other models. This dropout does not have an impact on the malware classification performance.

The performance accuracy on all the individual feature sets using different data analytics models is shown in Fig. 2. The static "PE import" feature set consistently achieved better accuracy than all the other datasets performance for all the DL models used in our analysis and SVM technique. The number of features in the "PE import" feature set enables these models to learn the data patterns and adapt the DL weights to achieve better accuracy. We can also see that CNN and DNN1...DNN4 has slightly improved accuracy in comparison with LSTM1...LSTM4 models. It is also evident that the classical SVM did perform well and achieved comparable accuracy with CNN and DNN models for the "PE import" feature set. The dynamic "PE API" feature set accuracy performance is slightly closer to the PE Import function feature set. But, for the LSTM1 model, "PE API" feature set performed worse than the other two "PE section" and "PE image" feature sets. We used the four hyperparameters "number of units" for the LSTM1 model. The "PE section" and "PE image" feature sets are least performed in terms of accuracy, and it is clear that the "PE section" performed better than the "PE image" when using all the DNN models. It is interesting to see that the "PE section" and "PE image" equally performed well with the CNN model, but when we considered the CNN-LSTM model, the "PE image" set accuracy improved over the CNN model performance, and the "PE section" accuracy drastically reduced compared to CNN model. Overall, we can see that most of the individual feature sets performed well using CNN models and maintained consistency in all the four feature sets compared with other models .

Figs. 3 and 4 represents the macro and weighted average precision for the SVM and DLL models. These figures clearly show that all the four feature sets precision curve follows the accuracy performance curve trends shown in Fig. 2. Since macro and weighted precision performance is quite similar in almost all the models used in our experiments, we can infer that there is no bias towards either legitimate or malware class while classifying the malware using SVM and DL models. Additionally, the selected proportions of malware and legitimate samples are intact. It is also worthy to note that the false positive tuning efforts required for the individual four datasets time in decreasing order as follows, i.e, "PE section", "PE image", "PE API", and "PE import", based on the effective performance of the CNN model.

Figs. 5 and 6 illustrates the percentage of the actual malware classified as malware by SVM and DL models for the four individual

Table 7
Hyperparameters used in the DL models for performance evaluation.

Parameters	DNN1	DNN2	DNN3	DNN4	CNN	CNN-LSTM	LSTM1	LSTM2	LSTM3	LSTM4
Dense	1024,1	1024,768,1	1024,768,512,1	1024,768,512,1	128,1	1	1	1	1	1
Dropout	0.01	0.01,0.01	0.01,0.01,0.01,0.01	0.01,0.01,0.01,0.01	0.5	0.1	0.1	0.1,0.1	0.1,0.1,0.1	0.1,0.1,0.1,0.1
Activation	ReLU	ReLU,ReLU	ReLU,ReLU,ReLU	ReLU,ReLU,ReLU,ReLU	ReLU, ReLU	ReLU	–	–	–	–
Batchsize	64	64	64	64	–	–	32	32	32	32
epochs	200	200	200	200	200	200	200	200	200	200
Poolsize	–	–	–	–	2	2	–	–	–	–
Kernelsize	–	–	–	–	3	3	–	–	–	–
Number of Filters	–	–	–	–	64	64	–	–	–	–

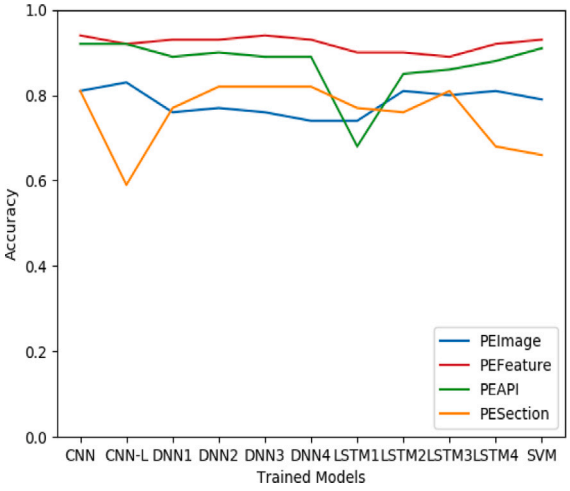


Fig. 2. Comparison of Accuracy on All models and four datasets.

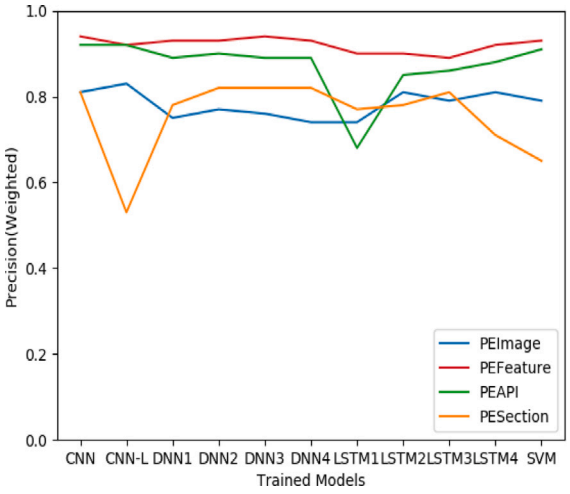


Fig. 3. Comparison of the Macro Precision for all the models.

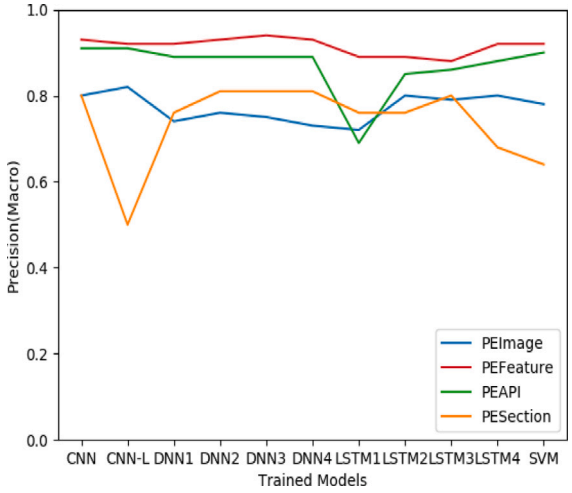


Fig. 4. Comparison of the Weighted Precision for all the models.

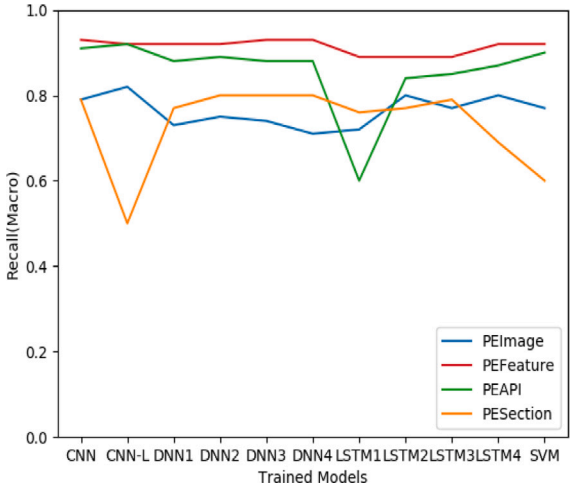


Fig. 5. Comparison of the Macro Recall for all the models.

feature datasets evaluation. We can see that the "PE import" dataset can identify more actual malware numbers from the datasets than the other three sets for all the models considered in our evaluation . This could indicate that extracting the static importing function calls features from the binary samples may not overlook the malware and benign static data differences in samples. Further, the "PE API" call sequence extracted from datasets using dynamic analysis performed slightly closer to the "PE import" results. Based on these two feature sets' performance, we can construe that the interaction with the system core modules using API functions calls in windows is clearly an important aspect to be considered for distinguishing the malware and benign file. It is also evident that "PE API" datasets accurate malware sample detection improved as the number of hidden layers increased in the LSTM model. Additionally, the LSTM4 model showed closely comparable performance with the CNN model. The Fig. 5 also concludes that the least recall performance achieved among the four feature sets are "PE image" and "PE section" in almost all the models except LSTM1.

In general, there may be a trade-off between precision and recall metrics when we train the model. Depending on the business requirement of the application, either false positives or false negatives for the trained model can be neglected. For instance, in malware detection, even a single malware detection miss may have a catastrophic impact on the organization. So, we always make sure the false negative should be as low as possible. On the other hand, false positive can be overlooked to a certain extent, in particular, when the organization has the

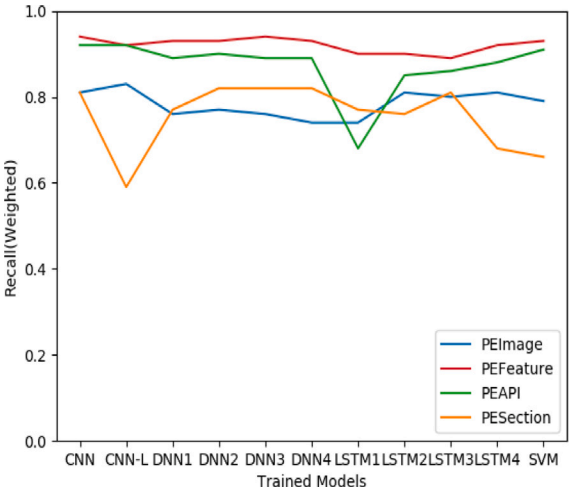


Fig. 6. Comparison of the Weighted Recall for all the models.

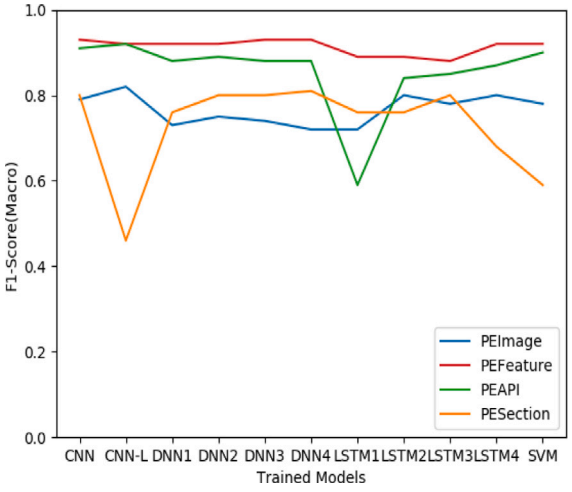


Fig. 7. Comparison of the Macro F1-Macro for all the models.

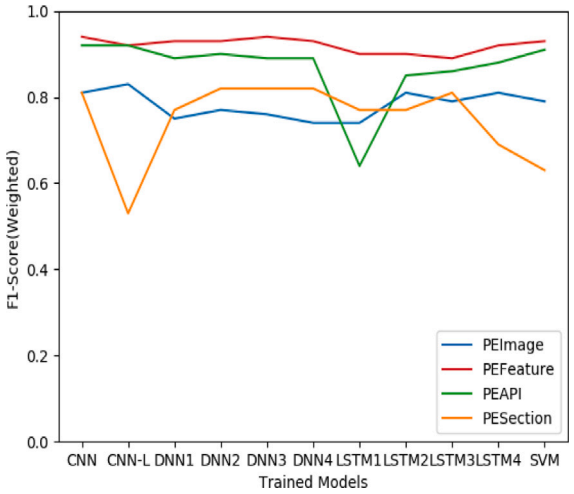


Fig. 8. Comparison of the Weighted F1-Score for all the models.

Table 8
Best performed model metrics (0:Benign and 1:Malware).

Best model	Class	Precision	Recall	F1-score
PE Images				
CNN-LSTM	0	0.76	0.78	0.77
	1	0.87	0.85	0.86
PE API				
CNN	0	0.91	0.87	0.89
	1	0.92	0.94	0.93
PE Features				
CNN	0	0.91	0.92	0.92
	1	0.95	0.94	0.95
PE Section				
DNN4	0	0.78	0.73	0.75
	1	0.84	0.87	0.86

Table 9
Classification report for the ImageApi (0:Benign and 1:Attack).

	Precision	Recall	F1-Score
0	0.85	0.86	0.85
1	0.91	0.91	0.91
Macro avg	0.88	0.88	0.88
Weighted avg	0.89	0.89	0.89

workforce to analyze these alerts. However, more false positives can also be not tolerated and needed further retraining of the models. F1-score can actually be helpful to analyze the balance between precision and recall. As illustrated in Figs. 7 and 8, The F1-Score for all the four individual feature datasets determined in all the models for our performance evaluation shows that the balance between the precision and recall is obtained. The F1-Score line graphs followed closely the same trend as the precision and recall line graphs, and this shows that the false positives and false negative numbers are almost the same, at least for the majority of the model result cases. As discussed in Figs. 3 and 4 description, the macro and weighted average results shown in Figs. 7 and 8 are closely comparable in all the models for the four individual feature datasets.

Table 8 represents the best performance models and their performance metrics to classify the test samples as benign or malware for each feature set case. We can see that CNN gives the best performance for the "PE import", "PE API", and partly used CNN in "PE image" for malware classification in both benign and malware identification. The "PE import" feature set achieved precision and recall performance 95% and 94% respectively for malware class and 91% and 92% in benign class classification. On the other hand, the best performance among the models for the least performed "PE section" dataset in all the four individual datasets is achieved using the DNN4 model. The precision and recall performance for the DNN4 model is 84% and 87% in the case of malware class and 78% and 73% in benign class classification. Overall, we can conclude that the CNN model is considered the best model to get expected performance results in any four individual dataset evaluations.

To evaluate our feature fusion approach, we have chosen the combinations of the four individual feature datasets. The idea is to cover the different views of these feature sets by selecting two or three or all combinations of the static, dynamic, and image feature datasets and evaluate the performance of these combined features using the best performed DL model CNN. So, our combination of feature sets include "PE image" and "PE API" (ImageAPI); "PE import", "PE section", and "PE API" (ImportSectionAPI); "PE import" and "PE section" (ImportSection); "PE image", "PE import", and "PE section" (ImageImportSection); "PE image", "PE import", "PE API" and "PE section" (Mergeall). The classification reports are produced for all these fusion feature set combinations using the CNN model for comparative analysis.

Table 9 represents the performance classification report for combined "PE Image" and "PE API" features. From Table 9, although this fusion feature set may not perform better than the best performed

Table 10
Classification report for the "ImportSection" (0:Benign and 1:Attack).

	Precision	Recall	F1-Score
0	0.91	0.94	0.92
1	0.96	0.95	0.95
Macro avg	0.94	0.94	0.94
Weighted avg	0.94	0.94	0.94

Table 11
Classification report for the ImportSectionApi (0:Benign and 1:Attack).

	Precision	Recall	F1-Score
0	0.95	0.94	0.94
1	0.96	0.97	0.97
Macro avg	0.96	0.95	0.96
Weighted avg	0.96	0.96	0.96

Table 12
Classification report for the ImageImportSection (0:Benign and 1:Malware).

	Precision	Recall	F1-Score
0	0.89	0.95	0.92
1	0.97	0.93	0.95
Macro avg	0.93	0.94	0.93
Weighted avg	0.94	0.93	0.93

individual feature set "PE import", the performance of this fusion set drastically improved in comparison with the least performed individual feature set "PE image". This performance improvement is due to combining the "PE API" feature set with "PE image". Overall, the macro and weighted average precision and recall for feature combination "ImageAPI" achieved 89% and 89%, respectively, which is much better than "PE image" metrics and slightly comparable to "PE API" features.

Another two feature combinations, i.e, "PE import" and "PE section" are clubbed together, and applied this combination on the CNN model yields the performance results as depicted in Table 10. The precision and recall for the dataset malware and benign classes on the "ImportSection" feature set show that "ImportSection" yields slightly better results in comparison with the individual feature set, either "PE section" or "PE import". The "ImportAPI" performed the same as the "PE API" dynamic feature in terms of accuracy . Furthermore, the "ImageAPI" fusion set average macro shows that the "ImageAPI" slightly improved the performance in comparison with "PE import" or "PE API" set. On the other side, the weighted average is the same for both the fusion set and the individual feature and section feature sets. Overall, this result shows that the fusion feature slightly improved some performance metrics in comparison to the "PE import" or "PE section" features.

Table 11 illustrates the performance classification report results for the fusion set combination of "PE image", "PE import" and "PE section features". As shown in the Table 11, the malware and benign class classification Precision, Recall, and F1-Score for the fusion set of 3 feature combinations "ImportSectionAPI" outperformed the two feature combinations "ImportSection" and "ImageAPI" for both the macro and weighted average cases. This clearly shows that our hypothesis on multi-view of the features can enhance the performance and improve the accurate classification ability for malware classification. In addition, we can also see that "ImportSectionAPI" feature set combination performance significantly improved compared to the best individual feature set "PE import" evaluated on the best performed CNN model among all the models considered for our experiments.

The combination of the "PE image", "PE import", and "PE section" feature sets is also considered as one of the fusion sets for our evaluation. Table 12 shows the performance classification report for the "ImageImportSection" fusion feature set. Even though the overall performance of the "ImageImportSection" is not better than "ImportSectionAPI", the precision value for the malware class is slightly higher than the three fusion set "ImportSectionAPI". This shows that different

Table 13
Classification report for the MergeAll (0:Benign and 1:Malware).

	Precision	Recall	F1-Score
0	0.93	0.93	0.93
1	0.96	0.96	0.96
Macro avg	0.94	0.95	0.95
Weighted avg	0.95	0.95	0.95

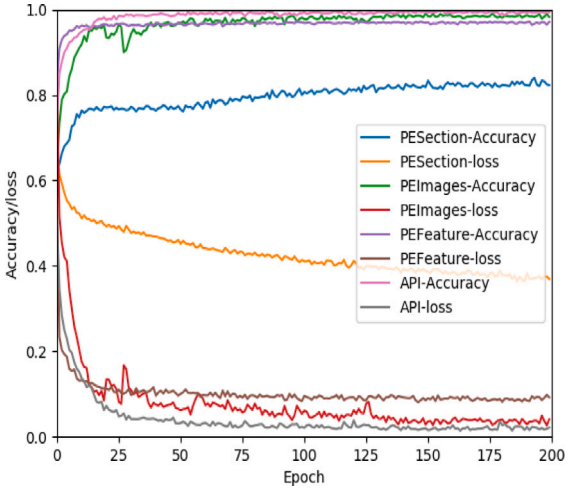


Fig. 9. Accuracy/loss characteristics for the individual four feature sets.

views of feature combinations may interpret unseen improved results. So, selecting the combination of the features is key to obtaining superior results. This selection may depend on the number of features, dataset size, how the features are extracted, and the expectation of the performance model and model used for testing.

We have also considered merging all the feature set combinations to evaluate the feature fusion approach for malware classification. Table 13 presents the classification report results for the "Mergeall" feature combination to classify the benign and malware using the CNN model. The results show that the "Mergeall" feature set achieved better performance than the best performed individual feature set, "PE import". These results support the previous claims that feature fusion sets could perform better than individual feature sets as input features for DL-based malware classification. It is also evident that "Mergeall" comfortably performed better than our evaluation's two set feature fusion combinations . However, one of the three set combinations, "ImportSectionAPI" slightly performed better than "Mergeall". We believe that the addition of the feature set "PE image", which is seen to be almost least performed in individual feature set evaluations, to the "ImportSectionAPI" feature may incur the misclassification of a few samples and hence "Mergeall" overall performance not better than feature fusion set "ImportSectionAPI".

Fig. 9 illustrates the training accuracy and loss performances for the four individual feature sets when the epoch varies from 1 to 200 using the CNN model. As shown in the Fig. 9, The "PE API" feature set achieved training accuracy of almost 99% when the epoch reached 70, followed by swinging around 98% and 100% until the epoch reaches 100 and then settled down to 100%. The "PE image" feature set achieved 96% accuracy when the epoch was around 65 and then fluctuate between 97% and 98% until the 200 epoch was completed. The "PE import" feature set accuracy quickly converges to 95% nearly at epoch 15 and then steadily maintained the same accuracy until complete all the epochs. In contrary to those three feature sets, the "PE section" feature set training accuracy is poor and able to achieve well below 80% by the end of all epochs. The few features in "PE section" have made it difficult to achieve better training accuracy. The loss curves for all these four feature sets follow the downwards

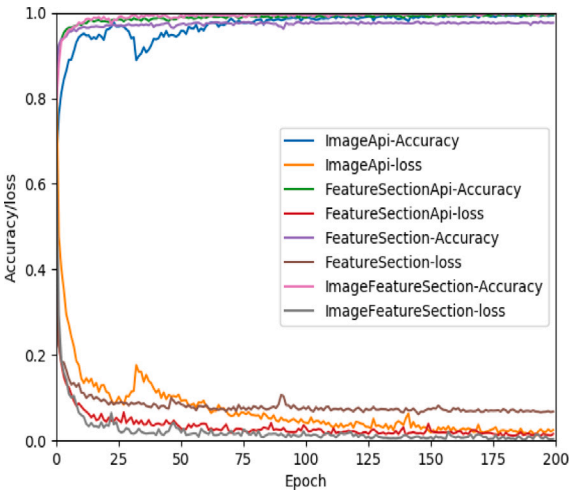


Fig. 10. Accuracy/loss characteristics for the four fusion feature sets.

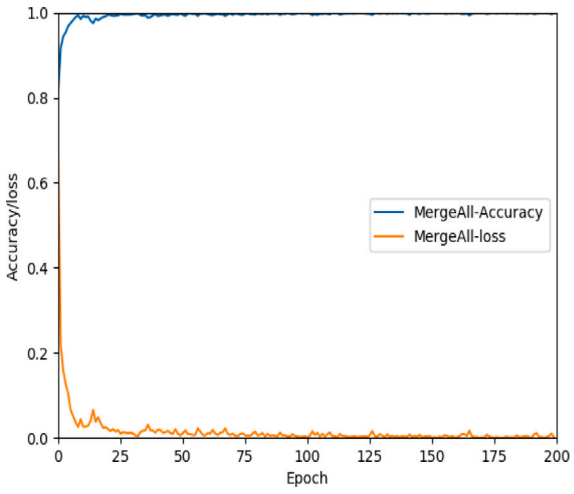


Fig. 11. Accuracy/loss characteristics for the best performed feature set.

trend in accordance with the accuracy achieved by those feature sets. Overall, we can conclude that the "PE API" and "PE import" feature sets are trained with good accuracy and this can be an indication of these feature sets did perform well on the test dataset. However, the performances of the proposed models on "PE image" can be enhanced by including the CNN-based pretrained models [15]. In the future, we are planning to create a big malware data of "PE image" and employ a CNN-based pretrained model to achieve optimal performance. Since the number of data samples is less, the CNN based-pretrained models are not explored in this study.

The Fig. 10 depicts the training accuracy and loss curves for the two and three feature set fusion combinations when the CNN model is applied to the training datasets. We can see that the three feature fusion sets "ImageFeatureSection" and "FeatureSectionAPI" converged faster to achieve better training accuracy than the two feature fusion sets "ImageAPI" and "FeatureSection". For instance, The three feature fusion sets achieved 100% when the epoch reached 60, and then "ImageFeatureSection" steadily maintained afterward until the epoch reached 200. At the same time, "FeatureSectionAPI" had some downward spikes until 125 before closing 100% at epoch 200. But, the two feature sets accuracy never crosses 96% until epoch 75 and then settles down to maintain steady accuracies. Overall, these results could support our earlier discussion on three fusion feature sets achieving better performance than the two fusion feature sets.

Fig. 11 shows the "Mergeall" case feature fusion set training accuracy/loss characteristics when processed through the CNN model. The training accuracy was achieved to 100% within the five epoch iterations and finally settled down later around epoch 25 to provide 100% accuracy. Overall, based on the all Figs. 9–11, the fusion feature sets can achieve better training accuracy and even the best accuracy performance when more number of the feature sets combined, which are representing a particular view of the malware or benign sample. Overall, we can conclude that the "Mergeall" case is the best performed feature fusion set in our performance evaluation.

The complexity of the DL models can be compared using model parameters generated during the model training. Fig. 12 presents the comparison of the complexity and the performance of the different DL models used for the evaluation of the four individual feature sets and five feature fusion sets in our approach. The DL models DNN, LSTM, CNN, and CNN-LSTM were considered for our extensive performance evaluation of the feature sets. The DNN4 and LSTM4 cases are only considered from the DNN and LSTM models to clarify the figure representation and easy comparison of different models.

We plotted the ROC characteristics for the selected CNN and classical SVM models by inputting the "Mergeall" fusion feature set. As shown in Fig. 13, the DL CNN model and the classical ML model achieved comparable performance, and interestingly, both the models obtained Area Under Curve (AUC) value of 0.98. Thus, it emphasizes that the combination of feature sets taken from different aspects of the dataset samples would enable the classical SVM model to perform compared to the proposed CNN model. We also note that to what extent the fusion feature datasets can impact the performance of several ML models is out of the scope and can be considered as future work.

5.2. t-SNE feature visualization

t-SNE is helpful when we want to visualize the high dimensional data in two or three dimensions. In general, DL models act as a black box; we have limited visibility of the learning ability of hidden layers and visualization of the results for a better understanding of the DL model performance. We have obtained the penultimate layer features of the proposed CNN model in our feature fusion approach. These features are given as input to the t-SNE for the two-dimensional representation, and the obtained t-SNE plot is shown in Fig. 14. Fig. 14 indicates that most of the malware and benign sample data are formed in different clusters, and slight overlapping of the malware data with benign feature data can be seen in the plot. Overall, our model's clear distinction between the attack and benign feature data helps to achieve better performance, although the model is not achieved 100% accuracy.

5.3. Large-scale learning with single and meta-classifier

Computation time is one of the constraints of processing large-scale datasets. So, large-scale learning optimization algorithms are needed to obtain good results for large-scale datasets. We have leveraged the supervised learning algorithms Random Forest (RF) and SVM as the downward stream layer with the proposed CNN model. Large-scale CNN with RF achieved slightly better and comparable results with CNN with the SVM algorithm. So, the CNN with RF algorithm is presented here to compare other cases. Table 14a illustrates confusion matrix plot for the input feature fusion set "Mergeall" applied to the large-scale CNN with the RF algorithm. We can observe that the large-scale CNN with RF can classify 222 being files correctly out of 242 and 384 malicious files out of 400. Even though there is no significant performance improvement compared to the standalone proposed CNN model, the CNN with RF achieved comparable results and did not show noticeable performance degradation.

Table 14b represents the confusion matrix results for the large-scale stacked classifier. We have considered the RF classifier and RBF

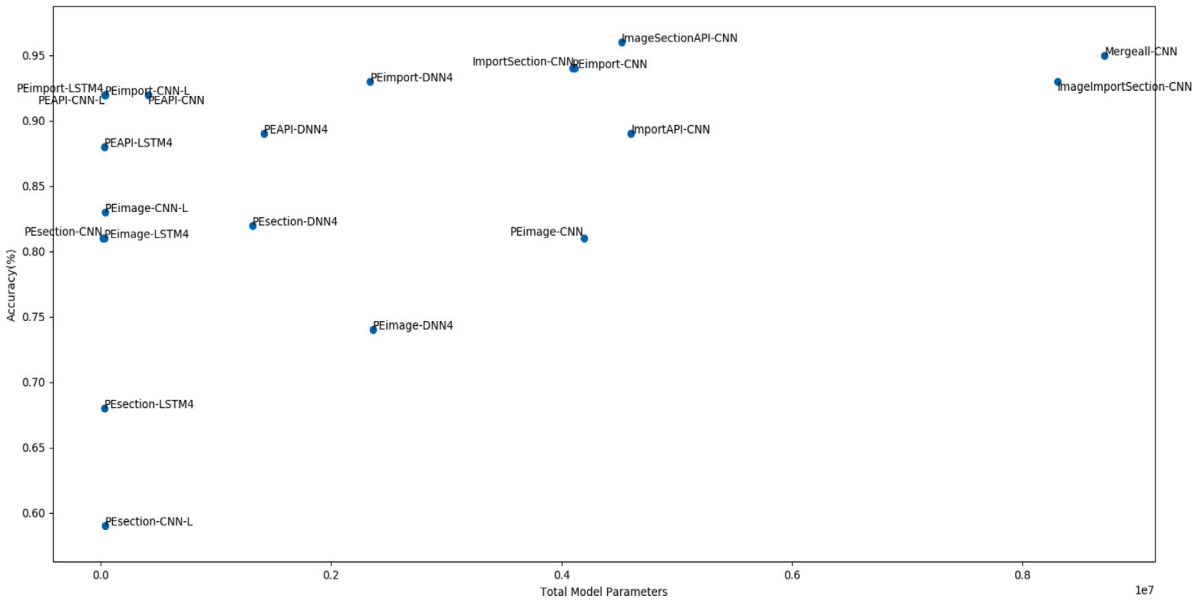


Fig. 12. Comparison of number of model parameters and accuracy of the DL models.

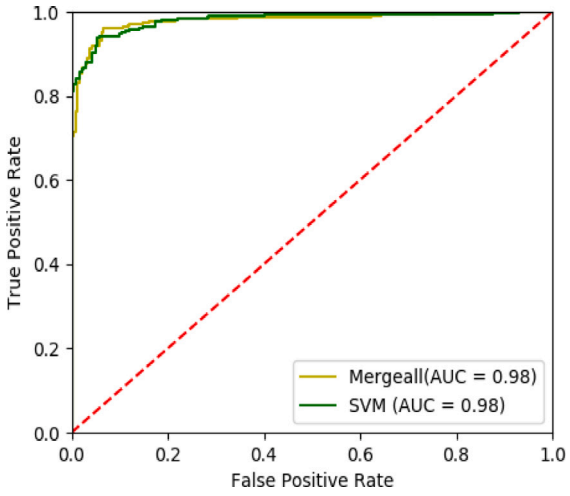


Fig. 13. ROC Characteristics for best performed model and classical SVM.

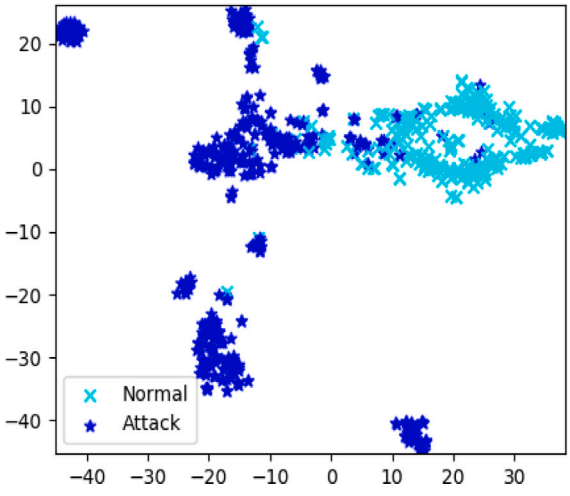


Fig. 14. t-SNE visualization of the best performed model for the best feature set.

kernel SVM as the initial estimators and the linear regression as the final estimator in our stacking classifier. From Table 14b, the stacked classifier was able to correctly classify 385 files as malware, which is slightly more than large-scale CNN with RF, CNN, and SVM results. On the other side, the benign file classification is slightly lower than the other cases used for comparison.

Tables 14c and 14d shows the confusion matrix results for the feature fusion sets "Mergeall" to the SVM and proposed CNN model. The CNN model correctly classified 384 malware files as malware, whereas SVM correctly classify 378 files as malware. On the other side, CNN correctly classified 225 benign files, while SVM correctly classified 222 benign files. It is clearly evident that the CNN model did perform well compared to the SVM classifier, in particular, considering the size of the malware dataset used. Overall, based on the results in Table 14, we can infer that both CNN and the classifiers in large-scale learning and stacked classifiers are disconnected. These two models can be connected together by proposing a new loss function, and this type of learning during training a model can enhance the performance of the malware detection model [44]. This is one of the future directions of the proposed work.

5.4. Generalization

With the aim to show that the proposed method is more generalizable, the performance of the proposed method is evaluated on other two unseen malware data samples such as sample1 and sample2. Each dataset contains 3,000 randomly selected unseen malware data samples. The Figs. 15 and 16 show the performance results of the proposed model CNN and classic machine learning model SVM malware classification on the two datasets samples. In Fig. 15, we can see that the Proposed model CNN was able to correctly classify 2,996 malware samples out of the 3,000 samples in dataset sample 1. On the other side, the SVM model can correctly classify only 2,214 malware samples as malware. The misclassification of the 786 malware samples as benign by the SVM is an indication that it is unacceptable performance for malware classification. Overall, we can construe that the proposed model outperformed the SVM model using our feature fusion approach.

The second dataset sample2 is applied to the proposed model and SVM under similar experimental settings. As shown in Fig. 16, the proposed model CNN performed better than the SVM results and was able to accurately classify 2,958 malware samples in the malware category out of the 3,000 malware samples. We can observe that the SVM

Table 14
The confusion matrix for the large-scale, meta-classifier and feature fusion best case (0:Benign, 1:Malware).

	0	1
0	222	19
1	17	384

(a) Large-scale Random Forest

	0	1
0	220	21
1	16	385

(b) Stacked classifier

	0	1
0	222	19
1	23	378

(c) Mergeall case - SVM

	0	1
0	225	16
1	17	384

(d) Mergeall case - CNN

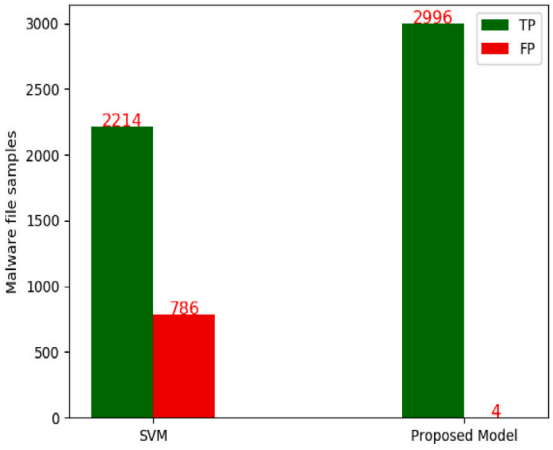


Fig. 15. SVM and CNN malware class classification on sample1 dataset.

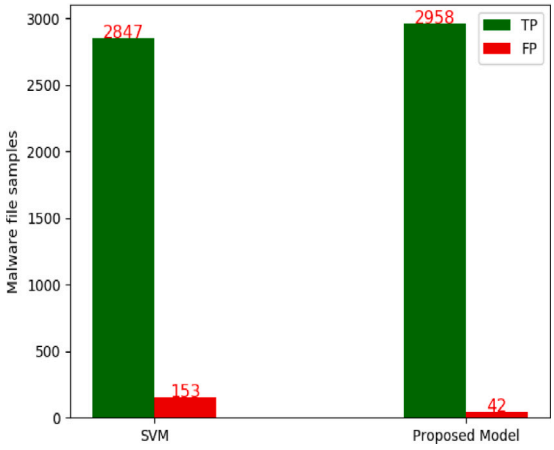


Fig. 16. SVM and CNN malware class classification sample2 dataset.

has generated 153 false negatives by misclassifying the malware as a benign fie, whereas CNN generated 42 false negatives on a set of 3,000 samples. Our performance results on these two malware data sample sets show that the proposed model CNN significantly performed well to accurately classify the malware, and SVM performed slightly better when using the input sample 2 dataset rather than the sample1 dataset. Furthermore, these malware classification results on the two malware data sample sets show that our proposed model is generalizable, robust, and able to classify unseen malware samples.

5.5. Advantages and limitations of the proposed approach

Our proposed DL CNN-based multi-view feature fusion approach for malware classification achieved better performance results in comparison with the results obtained from best performed individual feature set applied on the best performed CNN model for our evaluation. Thus, the feature fusion sets approach achieved an accuracy of 97% in the test dataset samples, whereas the best performed individual feature set obtained 94% accuracy. Although the percentage of the accuracy difference for the two feature sets is smaller, it is a considerable improvement in malware classification because an unidentified and stealthy malware successful compromise can have a catastrophic effect on the impacted organization's businesses.

Any particular view of the executable binary files to extract features can be bypassed and leveraged to evade malware detection using DL or ML models. An adversary could use simple tricks to modify the binary files for achieving detection mode evasion. For instance, An adversary can unpack the PE file using upx_unpack; move ".text" to ".xxx" with valid entry point for code instructions; create ".text" file and replace ".text" with ".text" from legitimate executable "calc.exe". These simple structural changes in the PE file can evade malware detection and also do not change the file's malicious behavior. In [45], the authors mention that these small static changes can fool the ML models to change the prediction from malware class to benign class. We believe that our feature fusion approach can easily defend evasion attempts by leveraging the features chosen from a single view of PE files. Hence, We have considered a combination of static, dynamic, and image features from PE files to tackle such attacks.

However, Our feature fusion approach requires a cautious selection of features from static, dynamic, or image analysis-based on a number of factors such as sample size, operating in a production or test environment, available hardware resources, and workforce domain knowledge. Additionally, feature extraction can be a tedious and time-consuming process and in particular, for dynamic analysis, a virtual environment isolated from the internet can be required. So, it is important to choose the minimal number of features that could cover the different aspects of malware characteristics and behavior to combat the adversarial effects.

The proportion of the malware and benignware file samples considered for our study does not represent the real-time settings. The number of available windows benignware samples is much higher than the malware samples in the real world. We used the benignware samples taken from the site portableapps.com and the 32-bit Windows 7 ultimate directory. The total number of available benignware from these two resources is minimal. So, the proportion of the malware samples is higher

than the benignware in our dataset. We plan to collect more windows PE file benignware samples and validate our multi view feature fusion models and perform large-scale dataset validation, which resembles the real-world production environment malware binary classification setup. One of our future works is to perform the experiments using the imbalanced large-scale datasets with more benignware sample files, and fewer number of malware sample files.

Our study is focused on the malware binaries observed in Windows environments, precisely, limited to PE format file malware classification and analysis. So, the outcomes of our work may or may not be applicable to Unix-based ELF executable or IoT malware samples. Thus, we leave the validation of the feature fusion approach to Unix-based malware as one of our future works.

6. Conclusion and future works

This paper proposed a deep learning CNN model for effective malware classification using our feature fusion set approach. The proposed CNN model was selected by performing a comprehensive performance evaluation of the classic ML classifier SVM and DNN, CNN, and LSTM model architectures for our feature fusion approach. Our comparative performance analysis showed that the proposed CNN model outperformed other models in the majority of the individual feature datasets. The multi-view perspective of the executable binary file is considered to select the different f fusion feature sets for further investigation. Our experimental evaluation shows that the fusion feature sets performed better than the individual feature sets. Based on the majority of our evaluation result cases, if the number of features incorporated in the fusion feature set increases, then the performance of the fusion feature set improves the performance for the same proposed deep learning model.

We are seeing that adversarial attacks on ML or DL models can have a significant impact on malware classification performance [46]. One of our future work is to validate the effectiveness of combined feature sets to defend the adversarial attacks. We also plan to investigate the best suited multi-view feature sets, which require less effort to extract those features and supports effective classification of sophisticated malware.

CRediT authorship contribution statement

Rajasekhar Chaganti: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Validation. **Vinayakumar Ravi:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Validation. **Tuan D. Pham:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

References

[1] Johnson J. Annual number of malware attacks worldwide from 2015 to 2020. Statista; 2021, https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/.

[2] Jovanović Bojan. A not-so-common cold: Malware statistics in 2021. Dataprot; 2021, https://dataprot.net/statistics/malware-statistics/.

[3] Gibert D, Mateu C, Planes J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. J Netw Comput Appl 2020;153:102526.

[4] Kolosnjaji B, Eraisha G, Webster G, Zarras A, Eckert C. Empowering convolutional networks for malware classification and analysis. In: 2017 International joint conference on neural networks. IEEE; 2017, p. 3838–45.

[5] Amer E, El-Sappagh S, Hu JW. Contextual identification of windows malware through semantic interpretation of API call sequence. Appl Sci 2020;10(21):7673.

[6] Huang X, Ma L, Yang W, Zhong Y. A method for windows malware detection based on deep learning. J Signal Process Syst 2021;93(2):265–73. http://dx.doi.org/10.1007/s11265-020-01588-1.

[7] Nisa M, Shah JH, Kanwal S, Raza M, Khan MA, Damaševičius R, Blažauskas T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. Appl Sci 2020;10(14):4966.

[8] Choi S, Bae J, Lee C, Kim Y, Kim J. Attention-based automated feature extraction for malware analysis. Sensors 2020;20(10):2893.

[9] Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Venkatraman S. Robust intelligent malware detection using deep learning. IEEE Access 2019;7:46717–38.

[10] Jain M, Andreopoulos W, Stamp M. Convolutional neural networks and extreme learning machines for malware classification. J Comput Virol Hacking Tech 2020;16(3):229–44.

[11] Sun G, Qian Q. Deep learning and visualization for identifying malware families. IEEE Trans Dependable Secure Comput 2018.

[12] Raff E, Barker J, Sylvester J, Brandon R, Catanzaro B, Nicholas CK. Malware detection by eating a whole exe. In: Workshops at the thirty-second AAAI conference on artificial intelligence. 2018.

[13] Vinayakumar R, Soman KP. DeepMalNet: evaluating shallow and deep networks for static PE malware detection. ICT Express 2018;4(4):255–8.

[14] Venkatraman S, Alazab M, Vinayakumar R. A hybrid deep learning image-based analysis for effective malware detection. J Inf Secur Appl 2019;47:377–89.

[15] Vasan D, Alazab M, Wassan S, Safaei B, Zheng Q. Image-based malware classification using ensemble of CNN architectures (IMCEC). Comput Secur 2020;92:101748.

[16] Cui Z, Xue F, Cai X, Cao Y, Wang GG, Chen J. Detection of malicious code variants based on deep learning. IEEE Trans Ind Inf 2018;14(7):3187–96.

[17] Ahmadi M, Ulyanov D, Semenov S, Trofimov M, Giacinto G. Novel feature extraction, selection and fusion for effective malware family classification. In: Proceedings of the 6th ACM conference on data and application security and privacy, 2017. 2016, p. 183–94.

[18] Ni S, Qian Q, Zhang R. Malware identification using visualization images and deep learning. Comput Secur 2018;77:871–85. http://dx.doi.org/10.1016/j.cose.2018.04.005.

[19] Kolosnjaji B, Zarras A, Webster G, Eckert C. Deep learning for classification of malware system call sequences. In: Australasian joint conference on artificial intelligence. Cham: Springer; 2016, p. 137–49.

[20] Catak FO, Yazı AF, Elezaj O, Ahmed J. Deep learning based sequential model for malware analysis using windows exe API calls. PeerJ Comput Sci 2020;6:e285.

[21] Abusitta A, Li MQ, Fung BC. Malware classification and composition analysis: A survey of recent developments. J Inf Secur Appl 2021;59:102828.

[22] Aslan ÖA, Samet R. A comprehensive review on malware detection approaches. IEEE Access 2020;8:6249–71.

[23] Schultz Matthew G. Feature extraction. 2001, https://www.fsl.cs.stonybrook.edu/docs/binaryeval/node4.html. [Accessed 20 June 2021].

[24] Saxe J, Berlin K. Deep neural network based malware detection using two dimensional binary program features. In: 2015 10th International conference on malicious and unwanted software. IEEE; 2015, p. 11–20.

[25] Azeez NA, Odufuwa OE, Misra S, Oluranti J, Damaševičius R. Windows PE malware detection using ensemble learning. In: Informatics, vol. 8, no. 1. Multidisciplinary Digital Publishing Institute; 2021, p. 10.

[26] Li C, Zheng J. API call-based malware classification using recurrent neural networks. J Cyber Secur Mobil 2021;617–40.

[27] Zhang Z, Qi P, Wang W. Dynamic malware analysis with feature engineering and feature learning. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34, (01). 2020, p. 1210–7.

[28] Burnap P, French R, Turner F, Jones K. Malware classification using self organising feature maps and machine activity data. Comput Secur 2018;73:399–410.

[29] Huang W, Stokes JW. Mtnet: a multi-task neural network for dynamic malware classification. In: International conference on detection of intrusions and malware, and vulnerability assessment. Cham: Springer; 2016, p. 399–418.

[30] Rhode M, Burnap P, Jones K. Early-stage malware prediction using recurrent neural networks. Comput Secur 2018;77:578–94.

[31] Appice Annalisa, Andresini Giuseppina, Malerba Donato. Clustering-aided multi-view classification: a case study on android malware detection. J Intell Inf Syst 2020;55(1):1–26.

[32] Millar Stuart, et al. Multi-view deep learning for zero-day android malware detection. J Inf Secur Appl 2021;58:102718.

[33] Darabian Hamid, et al. A multiview learning method for malware threat hunting: windows, IoT and android as case studies. World Wide Web 2020;23(2):1241–60.

[34] Haddadpajouh H, Azmoodeh A, Dehghantanha A, Parizi RM. MVFCC: A multi-view fuzzy consensus clustering model for malware threat attribution. IEEE Access 2020;8:139188-139198.

[35] Sahoo D. Cyber threat attribution with multi-view heuristic analysis. In: Handbook of big data analytics and forensics. Cham: Springer; 2022, p. 53–73.
[36] Chaganti R, Ravi V, Pham TD. Deep learning based cross architecture internet of things malware detection and classification. Comput Secur 2022;102779.
[37] Kyadige A, Rudd EM, Berlin K. Learning from context: A multi-view deep learning architecture for malware detection. In: 2020 IEEE security and privacy workshops. IEEE; 2020, p. 1–7.
[38] Shi W, Zhou X, Pang J, Liang G, Gu H. A new multitasking malware classification model based on feature fusion. In: 2018 2nd IEEE advanced information management, communicates, electronic and automation control conference. IEEE; 2018, p. 2376–81.
[39] Bai J, Wang J. Improving malware detection using multi-view ensemble learning. Secur Commun Netw 2016;9(17):4227–41.
[40] Oliveira Angelo. Malware analysis datasets: PE section headers. IEEE Dataport; 2019, http://dx.doi.org/10.21227/2czh-es14.
[41] Oliveira Angelo. Malware analysis datasets: Top-1000 PE imports. IEEE Dataport; 2019, http://dx.doi.org/10.21227/004e-v304.
[42] Oliveira Angelo. Malware analysis datasets: API call sequences. IEEE Dataport; 2019, http://dx.doi.org/10.21227/tqqm-aq14.
[43] Oliveira Angelo. Malware analysis datasets: Raw PE as image. IEEE Dataport; 2019, http://dx.doi.org/10.21227/8brp-j220.
[44] Huang FJ, LeCun Y. Large-scale learning with SVM and convolutional nets for generic object categorization. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, vol. 1. 2006, p. 284–91. http://dx.doi.org/10.1109/CVPR.2006.164.
[45] Anderson H, Kharkar A, Filar B, Roth P. Bot vs. bot : Evading machine learning malware detection why machine learning? BlackHat; 2017, https://github.com/EndgameInc/gym-malware.
[46] Kolosnjaji B, Demontis A, Biggio B, Maiorca D, Giacinto G, Eckert C, Roli F. Adversarial malware binaries: Evading deep learning for malware detection in executables. In: 2018 26th European signal processing conference. IEEE; 2018, p. 533–7.