electronics

MDPI

Article

Image-Based Malware Classification Using VGG19 Network and Spatial Convolutional Attention

Mazhar Javed Awan [1,*], Osama Ahmed Masood [1], Mazin Abed Mohammed [2], Awais Yasin [3], Azlan Mohd Zain [4], Robertas Damaševičius [5,*] and Karrar Hameed Abdulkareem [6]

1 Department of Software Engineering, University of Management and Technology, Lahore 54770, Pakistan; s2018266125@umt.edu.pk
2 College of Computer Science and Information Technology, University of Anbar, Ramadi 31001, Iraq; mazinalshujeary@uoanbar.edu.iq
3 Department of Computer Engineering, National University of Technology, Islamabad 44000, Pakistan; awaisyasin@nutech.edu.pk
4 UTM Big Data Centre, School of Computing, Universiti Teknologi Malaysia, Skudai 81310, Johor, Malaysia; azlanmz@utm.my
5 Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland
6 College of Agriculture, Al-Muthanna University, Samawah 66001, Iraq; khak9784@mu.edu.iq
* Correspondence: mazhar.awan@umt.edu.pk (M.J.A.); robertas.damasevicius@polsl.pl (R.D.)

Abstract: In recent years the amount of malware spreading through the internet and infecting computers and other communication devices has tremendously increased. To date, countless techniques and methodologies have been proposed to detect and neutralize these malicious agents. However, as new and automated malware generation techniques emerge, a lot of malware continues to be produced, which can bypass some state-of-the-art malware detection methods. Therefore, there is a need for the classification and detection of these adversarial agents that can compromise the security of people, organizations, and countless other forms of digital assets. In this paper, we propose a spatial attention and convolutional neural network (SACNN) based on deep learning framework for image-based classification of 25 well-known malware families with and without class balancing. Performance was evaluated on the Malimg benchmark dataset using precision, recall, specificity, precision, and F1 score on which our proposed model with class balancing reached 97.42%, 97.95%, 97.33%, 97.11%, and 97.32%. We also conducted experiments on SACNN with class balancing on benign class, also produced above 97%. The results indicate that our proposed model can be used for image-based malware detection with high performance, despite being simpler as compared to other available solutions.

Keywords: malware detection; image processing; convolutional neural network; spatial attention; transfer learning; deep learning; security

Citation: Awan, M.J.; Masood, O.A.; Mohammed, M.A.; Yasin, A.; Zain, A.M.; Damaševičius, R.; Abdulkareem, K.H. Image-Based Malware Classification Using VGG19 Network and Spatial Convolutional Attention. *Electronics* **2021**, *10*, 2444. https://doi.org/10.3390/electronics10192444

Academic Editors: Sang-Soo Yeo and Damien Sauveron

Received: 30 August 2021
Accepted: 6 October 2021
Published: 8 October 2021

1. Introduction

Malware (also known as malicious program) is malicious software developed intentionally to cause harm to computer systems. It is used to attack, infiltrate, or gain access to some digital assets that may be very sensitive in nature or cause damage or unwanted results from a system [1]. The primary purpose is to cause harm and invade assets whose access is not public. On average, 360,000 new malware files were detected every day in 2020, and the number of files found daily has increased by 5.2%. This rapid growth in malware production and distribution became possible due to the use of intelligent and automatic malware generation software such as SpyEye of Zeus and denial of service [2,3]. Newer dangers are evolving as blended threats continue to combine various types of assault into one with more deadly payloads. Despite the emergence of ground-breaking security defensive technologies, hacking, spoofing, phishing, and spyware are rising at an alarming rate. Moreover, phishing attacks are on the rise in many cases, leading to tricking the

recipient into clicking a malicious link that can lead to the installation of malicious software. The Internet of Things (IoT) is vulnerable to cyberattacks and malware originating from the Internet, which can result in data leakage, data manipulation, and substantial harm to society and people [4–6].

Malware attacks, which are often directed at traditional computers connected to the Internet, can also be directed at IoT devices and smart appliances. As a result, smart cybersecurity methods are required to safeguard millions of IoT users from harmful assaults. Malware detection over the years is carried out via different techniques. These range from extensive hand labeling to complex hybrid systems [7]. Anti-malware software use data mining techniques, information retrieval and information extraction, and association rule mining for analysis of malware, and adopting such techniques has enabled the production of new malware to skyrocket. The most popular techniques are the use of static and dynamic analysis for malware classification. Static analysis captures information from malware binaries without executing them, and dynamic analysis is carried out by observing the behavior of malware at run-time. Dynamic analysis is thought to be more reliable and efficient in the long term, but it had severe limitations. For example, it could not be deployed at the end-point in real-time since it takes too long for it to produce results since it requires time to analyze the malware during which the malware can fulfill its purpose [8–12].

However, a huge majority of these techniques rely on a large amount of feature engineering to function or require expert domain knowledge to build the features. This is a problem since the growth rate of the new malware is huge and this method cannot catch up to the speed of malware generation. Still, most of the well-known anti-malware software typically used the above-mentioned techniques along with a signature-based technique where a local database was created that stores signature patterns of the malware. For a long time, these ensembles of techniques were useful for fending off various types of malware attacks, but in recent years these techniques have been shown to be severely lacking in face of new automatic malware generation techniques, since they cannot keep up with the speed at which new malware can be produced [13,14].

Recently, machine learning (ML) models have been used in many domains. Moving on from this way, the most cutting-edge platforms use image processing along with help of machine learning or deep learning-based approaches for the classification of malware. A common solution is to use the ML algorithms and artificial neural networks (ANNs), which can be combined into more complex architectures such as using ensemble learning to identify malware from the features extracted from malware characteristics [15–21]. A popular choice has been the use of NNs along with SVM which were the most popular technique for malware classification adversarial attacks. For feature selection and classifier hyperparameter tuning, nature-inspired and metaheuristic optimization methods such as the 'Bat Algorithm' could be applied [22,23].

Lately, deep learning and image processing techniques are being used for malware classification problems [24] to overcome the intense feature engineering task or the requirement of domain knowledge. A technique, which was introduced in 2008 suggested that malware binaries can be converted into grayscale or RGB (Red, Green, Blue) color images that can be used to identify each type of malware specifically. Different types of malware of the same family have been observed to have identical image structures when converted from binary [25]. Recently, more complex NN architectures have been used; these are combinations of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) such as long short-term memory (LSTM) with other ML models as well as combinations of hybrid models [26–28].

This study aims to present a model that is the simplest solution for image-based malware detection which requires no special transformation of the images from binaries, no data augmentation or feature engineering, and no complex architecture. We aim to provide a state-of-the-art solution that uses cutting-edge methods to solve the malware recognition problem. Our study has the following contributions:

- To the best of our knowledge, this study is the first attention-based malware detection method working from 25 malware classification and benign class.
- The study proposes a transfer learning-based architecture that uses spatial convolutional attention to classify malware from multiple families through class weighting and without class balancing techniques.
- Lastly, we have performed extensive experiments testing using metrics such as precision, recall, F1- measure, confusion matrix ROC-AUC curves and produced above 97% result with and without class balancing.

The remaining sections are as follows: Section 2 presents an extensive literature review; Section 3 explains in detail the malware benchmark dataset used along with data pre-processing, and highlights the technologies used and their inherent limitations due to which we pivot towards more reliable methods (this section also includes the architecture of our proposed model); Section 4 describes experimental details such as setup, parametrization, and then discusses the results along with important visualizations; and Sections 5 and 6 present the discussion, limitations, and conclusions, where we describe the processes and highlight important details.

2. Related Work

Traditional signature and heuristic approaches to identify malicious software do not provide a sufficiently high degree of detection for new and previously undiscovered malware types. This decides whether the ML approaches can be used to solve this problem. Sophisticated deep learning methods combined with transfer learning techniques are used to improve the resilience and accuracy of malware detection without requiring advanced security understanding.

Rezende et al. [29] proposed a neural network architecture with transfer learning using ResNet-50. They used RGB images of size 224 $\times$ 224 with 10 folds with the Glorot uniform approach for weight initialization and Adam optimization. The model was trained for 750 epochs with a final accuracy of 98.62%. They also used GIST features with kNN using k = 4 resulting in an accuracy of 97.48% with bottleneck features producing an accuracy of 98.0%. Khan et al. [30] conducted an extensive analysis of transfer learning for malware classification using ResNet and GoogleNet with their data preparation pipeline and the top model. Resnet 18, 34, 50, 101, 152 achieved an accuracy of 83%, 86.51%, 86.62%, 85.94%, and 87.98%, respectively. The accuracy for GoogleNet was 84%.

Vasan et al. [31] used an ensemble model with VGG16 and ResNet-50. Both of these networks were fine-tuned. Along with PCA to reduce 90% of the features from the dataset and feed them into a one-vs-all multiclass SVM. They fine-tuned their model for 50 epochs and trained their CNN model from 100 to 200 epochs, achieving an accuracy of 99.50%. Yosinski et al. [32] proposed a model with 15 classes with 7087 examples with different types of feature extraction techniques producing their highest 97.47% accuracy. Nataraj et al. [33] used feature extraction with techniques such as GIST descriptors, and used ML algorithms such as KNN to produce an accuracy of 97%. Their algorithm uses static features classification and computed bi-gram distributions. This technique has the very basic flaw that if the adversary knows about their features, they can take countermeasures and avoid detection completely. Agarap et al. [34] used CNN or LSTM hybrid networks with SVM and other SVM hybrid architecture and deep learning models were proposed. Their CNN-SVM model stood at 77.22% accuracy, GRU-SVM stood at 84.92%, and the MLP-SVM hybrid model stood at 80.46% accuracy.

Akarsh et al. [35] proposed a CNN-LSTM hybrid model with the special transformation of the images. Their two-layer CNN which connected to an LSTM layer with 70 memory blocks and an FCN layer with 25 units with a softmax and categorical cross-entropy. The final accuracy on different splits was from 96.64% to 96.68%. In another paper, Akarsh et al. [36] used 2 layers of 1D CNN along with an LSTM for feature extraction with 0.1% dropout and 70 memory blocks of LSTM and a cost-sensitive algorithm to their model. They reported the highest accuracy of 95.5%. Sudhakar and Kumar [37] upgraded

the ResNet50 model by replacing the last layer of the model pretrained on ImageNet with a completely connected dense layer. The SoftMax layer receives the output of the fully connected dense layer for malware classification. Vinayakumar et al. [38] proposed Ember which used domain-level knowledge, different features from parsed portable execution (PE) files, and format-agnostic features like raw byte histogram.

Xiao et al. [39] proposed a malware classification framework (MalFCS) that visualized malware binaries as entropy graphs based on structural entropy. Then, deep CNNs were used to extract patterns shared by a family from entropy graphs. Finally, SVM was used to classify malware based on extracted features. Cui et al. [40] proposed to use the 'Bat Algorithm' for dynamic image resampling. Their purpose was to fight the imbalance in the dataset. Along with data augmentation, using this algorithm they were able to create a CNN with 94.5% accuracy.

In another study, Cui et al. [41] proposed a method for data equilibrium based on a NSGA-II genetic algorithm without equalization produced 92.1% accuracy, with a single objective algorithm produced 96.1% accuracy and with the multi-objective algorithm the highest accuracy of 97.1% was achieved. Jain et al. [42] used extreme learning machines (ELMs) with CNNs and proposed an ensemble model. They produced an accuracy of 96.30% with a single CNN layer and 95.7% with two CNN layers.

Naeem et al. [43] used an IOT based hybrid visualization technique with deep learning. By using different image ratios, they were able to develop models with accuracies up to 98.47% and 98.79% but were dependent on dynamic image features. Venkatraman et al. [44] proposed a hybrid architecture with a self-learning system. The proposed hybrid CNN BiLSTM and CNN BiGRU models and trained them with both cost-sensitive and cost-insensitive methods. All of their models with different types of parameters and settings range in accuracy from 94.48% to 96.3%.

Vu et al. [45] proposed a CNN-based architecture with transformations on the input images such as byte class, gradient, Hilbert, entropy, and hybrid image transformation with GIST and CNN-based models. Their GIST with grayscale images produced 94.27% accuracy and CNN performs the best with hybrid image transformation (HIT) technique. El-Shafai et al. [46] proposed a malware multi-classification framework that uses the pre-trained fine-tuned CNN models (AlexNet, DenseNet-201, DarkNet-53, ResNet-50, Inception-V3, VGG16, MobileNet-V2, and Places365-GoogleNet,) with transfer learning, while VGG16 has achieved the best performance for the malware recognition task.

Moussas and Andreatos [47] proposed a malware detection system based on a two-level ANN which used both file and image features. File features were used by the first-level ANN for classify malware, while the malware families creating a confusion were classified by a second level of ANNs using malware image features. Roseline et al. [48] used an ensemble deep forest method for malware identification and classification. Instead of relying on hand-crafted feature descriptors, the proposed method is data-independent and learns the discriminative representation from the data itself. Deep ensemble stacking and low model complexity distinguish the proposed method, which beats deep neural networks in identifying malware.

Verma et al. [49] suggested using a combination of the first-order and grey-level co-occurrence matrix (GLCM)-based second-order statistical texture features, which are classified using ensemble learning. The kernel-based ELM classifier was used for malware classification achieving 94.25% accuracy for the Malimg dataset. Çayır et al. [50] adopted the ensemble model of capsule networks (CapsNet). Instead of complex CNN architectures and domain-specific feature engineering techniques, the CapsNet model employs simple architecture engineering. Furthermore, CapsNet does not need transfer learning, and the model can be easily trained from scratch. Wozniak et al. [51] suggested using the developed RNN-LSTM classifier with the NAdam optimization algorithm for Android malware recognition. The performance evaluation on two benchmark datasets showed a 99% accuracy. Nisa et al. [52] propose a feature fusion technique for combining features derived from pre-trained AlexNet and Inception-v3 deep neural networks with features

extracted from images depicting malware code using segmentation-based fractal texture analysis (SFTA). SVM, KNN, decision tree (DT), and other classifiers are used to classify the characteristics retrieved from malware images. Hemalatha et al. [53] adopted the DenseNet model with a reweighted class-balanced loss function is used to gain substantial performance improvements in classifying malware images by dealing with unbalanced data difficulties.

Finally, Toldinas et al. [54] used a multistage deep learning image recognition approach for network intrusion detection. The network characteristics are converted into four-channel pictures (Red, Green, Blue, and Alpha). After that, the photos are utilized to train and evaluate the pre-trained deep learning model ResNet50. UNSW-NB15 and BOUN Ddos, two publicly available benchmark datasets, are used to test the technique.

In summary, the adoption of deep learning methods to recognize malware and network intrusions from features converted to images is currently on the rise, and a wide variety of neural network models and architectures are being explored, modified, and adopted. Nevertheless, considering a plethora of deep learning architectures available with a great number of hyper parameters more research is still needed to find the best solutions suitable for cybersecurity domain.

3. Materials and Methods

In this section, we describe all the technologies used and all the essential background information necessary to understand the proposed methodology.

3.1. Dataset

The malware image dataset, also called the Malimg dataset, is provided by Nataraj et al. [33]. The dataset contains 9389 grayscale images from 25 malware families. The Malimg dataset was used previously as a benchmark in numerous papers to evaluate malware detection methods, including the ones to be used in the IoT environments [43,55]. These are some of the well-known malware families and their different variants. These malware images were created from binaries of the malware as described by Conti et al. [25]. The conversion from binaries to images is done by first converting the binaries into 8-bit vectors and then these vectors are converted to grayscale images by taking each of the vectors as a pixel representing the intensity. Figure 1 illustrates the construction of the malware image from malware binary.

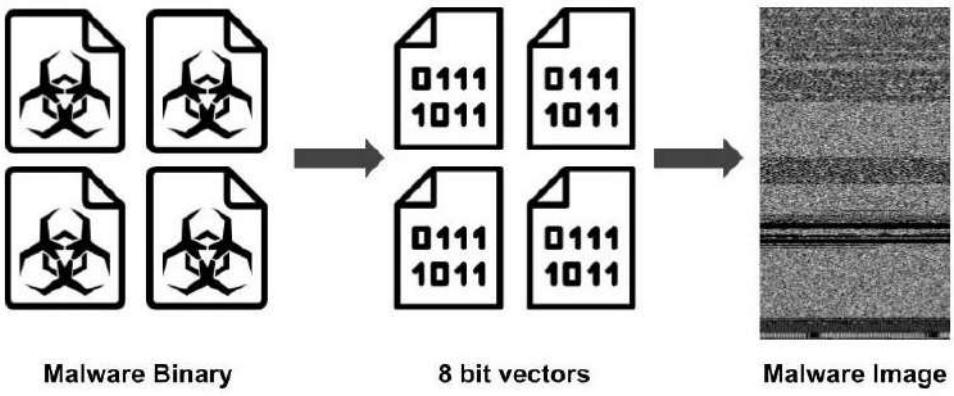
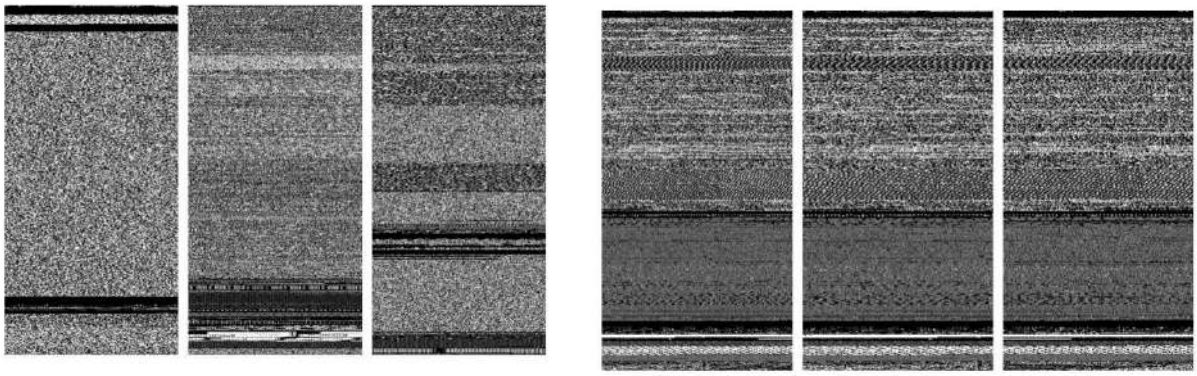


Figure 1. Process of creating a grayscale image from binary of a malware program.

It can be visually verified that these images created from binary files have very strong visual interclass relationships and, vice versa, differs from the images of unrelated classes, which assures us that the image-based classification is a very representative method for this dataset. Figure 2 shows representative grayscale images of malware.



(**a**) (**b**)

Figure 2. Grayscale images of malware: (**a**) unrelated classes of malware, and (**b**) related classes from the same malware family [33].

Another point of interest is that the malware images are very representative of the code format in the source binaries of the malware; by aligning the code file with the grayscale image we can observe that there is a high amount of correlation between the structure of the code file with the generated images from the dataset. Figure 3 shows a generated image from binary with its correlation.

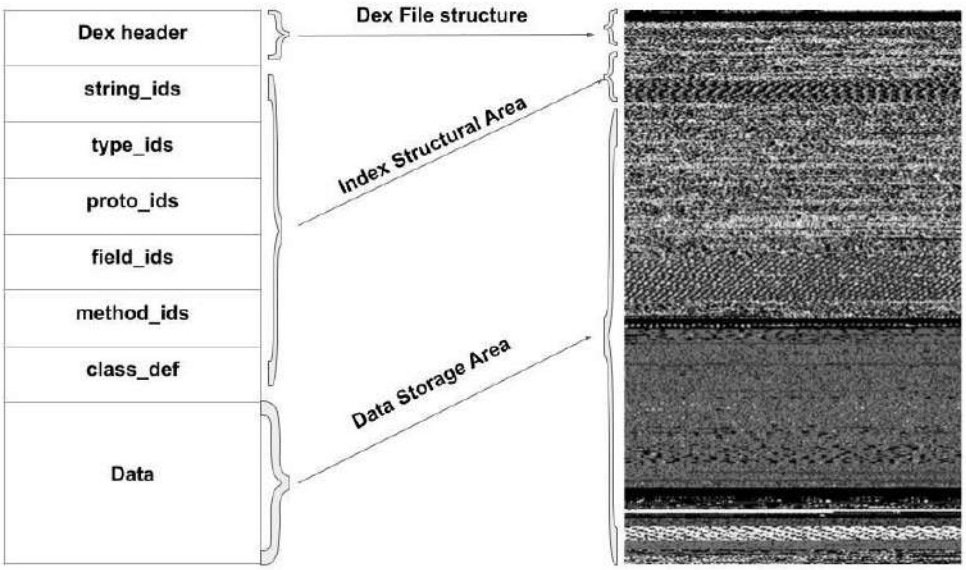


Figure 3. Sections of the image generated from binary that visually seem to have some correlation.

The details of the dataset containing 9389 images from 25 classes of malware along with their families are given along with frequency visualization. From the above table and histogram, it can be seen that the frequency of Allaple.A and its variant Allaple.L from the worm family of malware is the highest in the dataset with a few thousand examples. We also have Swizzor.gen!E from trojan family and Wintrim.BX from the Trojan downloader family, with less than a hundred examples. This shows that the dataset is highly unbalanced with wide gaps in the number of examples in each of the classes. Now we can further observe the dataset by using a scatter plot to visualize the different classes present in the dataset. Figure 4 shows the number of samples of malware families in the dataset.

To conclude, we have seen a few issues (namely the imbalance of the dataset and the structural correlation between binary and the image) about which we cannot comment further since the classification can produce very good results.

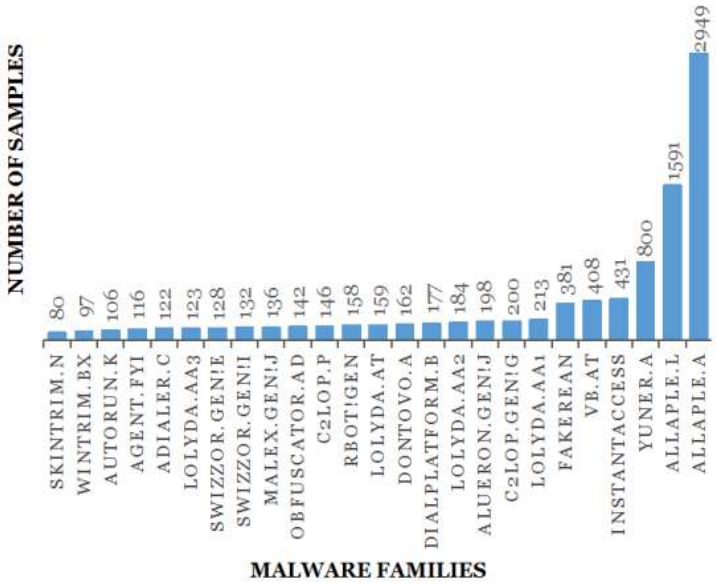


Figure 4. Visualization of the frequency of different variants of malware in the Malimg dataset.

3.2. Data Pre-Processing

The dataset was imported in its general image form which was transformed using a wavelet approach of transformation of binaries to images. We have used the Malimg dataset as described in the previous section. After simply importing the dataset from Kaggle we have used the TensorFlow implementation of Keras for importing and training our network in batches using the Keras flow, whereas the image size was set to be 224×224. Using the color mode of RGB, we have increased the channels of the images; this also creates a suitable batch for training due to which any complex preprocessing on simple images is not required. Other than the upscaling of the image from grayscale to RGB, no other transformation was applied to the dataset.

3.3. Techniques

3.3.1. Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of deep learning model that is inspired by the human visual cortex. They are a type of feed-forward neural network that has proven to be immensely successful in areas such as image processing and digital signal processing and other fields. These were the first models to have parameter sharing that made them ideal for tasks related to image processing and classification. The well-suited non-linearity in convolutional neural networks is the rectified linear unit (ReLU), which has proven to produce better results since it can resist a lot of problems. However, there is a problem with the CNN architecture. CNNs have a max-pooling layer inside them. Max pooling works by down sampling an image. It takes an image and from a fixed-sized window, it reduces whole frames to single values. The point to note is that this reduction is different from convolution since the values are not being merged to compute an output but are being ignored. This discarding of information is a very big challenge for CNNs since important information can be lost due to pooling [56–58]. Figure 5 shows the process of down sampling in the CNN model.

3.3.2. Spatial Attention Mechanism

The attention mechanism first became popular as an enhancement for encoder decoder-based neural machine translation systems. The mechanism of attention was first introduced by Bahdanau et al. [59] in 2015, when they used it for neural machine translation. They called it jointly learning by aligning to translate. In this paper, they called it soft alignment

and were able to produce a state of three art results. Figure 6 shows the attention mechanism in the LSTM model.

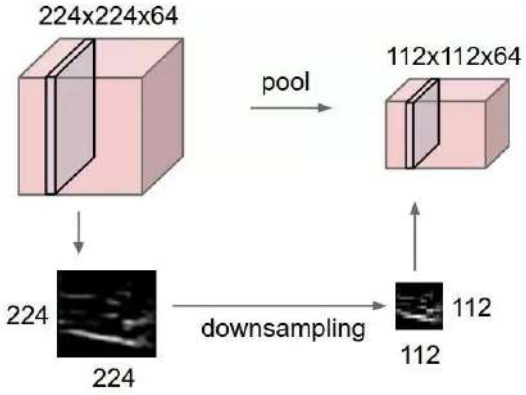


Figure 5. Down sampling inside a convolutional neural network with pooling layer.

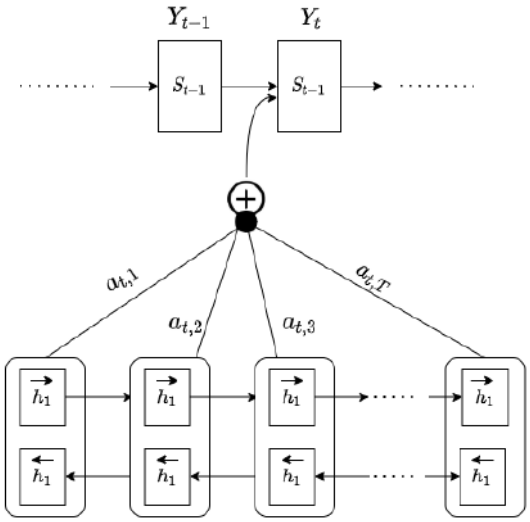


Figure 6. Long short-term memory generates attention from forwarding and backward hidden states.

Later Vaswani et al. [60] demonstrated how using local or global single head and multi-head attention can improve the results using very basic models. Since then, the attention mechanisms have been a big breakthrough in deep learning. These breakthroughs demonstrated how simply built mechanisms such as these can be used to fine grain and enhance the result of already state-of-the-art models. The idea behind attention was simple. It states that for tasks such as machine translation, the thing of importance is not only that all the input words are converted to context vectors, but that their relative importance is also considered; each word should be aligned with its relative importance. Attention can be classified into a few different types. Two major such types are global and local attention. Global attention is also known as soft attention. It is where all of the patches of a sequence or an image are given some weight. On the other hand, hard attention is also known as local attention and only gives weight to one patch of interest.

From this perspective, and to build on the previous discussion, we also explain how attention is used outside of sequential models. Attention has various types and forms that have been created over the span of a few years and include the use of attention in both simple and far complex ways. Here, a similar version of attention is used like the one used in Bahdanau et al. [59]. Another one is spatial transformer networks that use localization networks along with parameterized grid sampling and image sampling. This type of image processing is a very strong application of attention in the field of image

processing. More modern applications use residually connected architectures for tasks of this domain. Figure 7 shows the spatial attention process to enhance features before CNN.

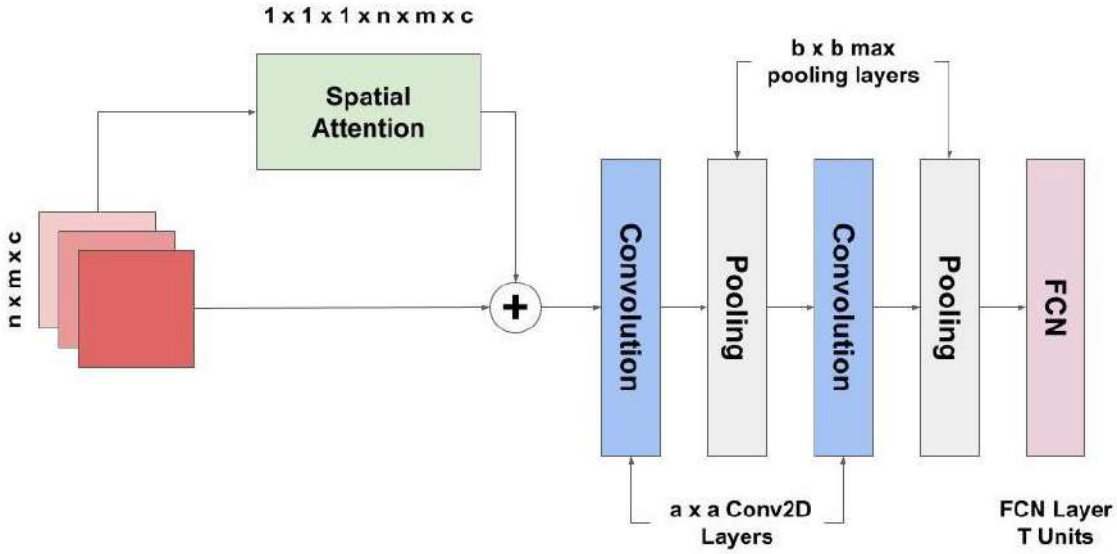


Figure 7. Process of spatial attention is added to enhance features before they are fed into CNN.

The importance and need for attention arise due to the fact that it can help overcome the information bottlenecks in deep learning models. Information bottlenecks are created when too much information is allowed to pass or process through a very narrow window where it becomes increasingly difficult to retain useful information by the network.

3.4. Our Proposed Architecture

The proposed model architecture consists of three main parts. The first part is a transfer learning model based on ImageNet called VGG19 [61]. The complete architecture of the model has been mentioned in greater detail. Here we only used the base of the VGG19 model, and the top part was removed. The base model has around 20,025,920 non trainable weights in it. To emphasize the fact that attention is better than most other approaches, we decided that it would be best if all the layers in the base VGG19 model were frozen. Since the layers were frozen the only purpose of the base VGG19 model was to behave as a feature extractor using convolution layers with the pre-trained weights. Using this architecture, it is possible to fine-tune and enhance the accuracy of the model, but we are refraining from doing so since we aim to show the enhancement of attention in our model and how it can perform much better than other models despite using a very simple form of attention mechanism. For a comparative analysis, we've only compared the results of VGG19. After analyzing the models, we have observed that the VGG19 model performs better on the malware dataset. Therefore, we are using VGG19 as a part of the proposed malware detection framework through transfer learning [62].

The next part of our model is the CNN model enhanced by attention. For this purpose, we have used a very simple form of attention. As we mentioned earlier, the model we introduce is for showing how a simple variant of attention mechanism can capture a lot of information and is a much simpler way of using attention. The technique that we have used here to generate attention is called dynamic spatial convolution. It is a type of spatial attention that works well for image processing and vision tasks. Dynamic spatial convolution uses a global average pooling mechanism which is easy to understand since in any given image all the areas are not equally important. Some specific regions are better suited for the task and are more useful [63].

Their spatial attention is generated using these normalized vectors and 2D convolutional layers. In the end, they are combined using a lambda layer. The lambda layer here

is used to rescale the GAP. The attention-enhanced feature maps are generated from this layer and are then fed into a dropout layer with a 25% dropout rate attached to a dense layer with 256 units. This is again passed through a 25% dropout layer for regularization and then a fully connected layer with 25 units. The activation used in this layer is SoftMax. Finally, Figure 8 visualizes our proposed architecture of the neural network.

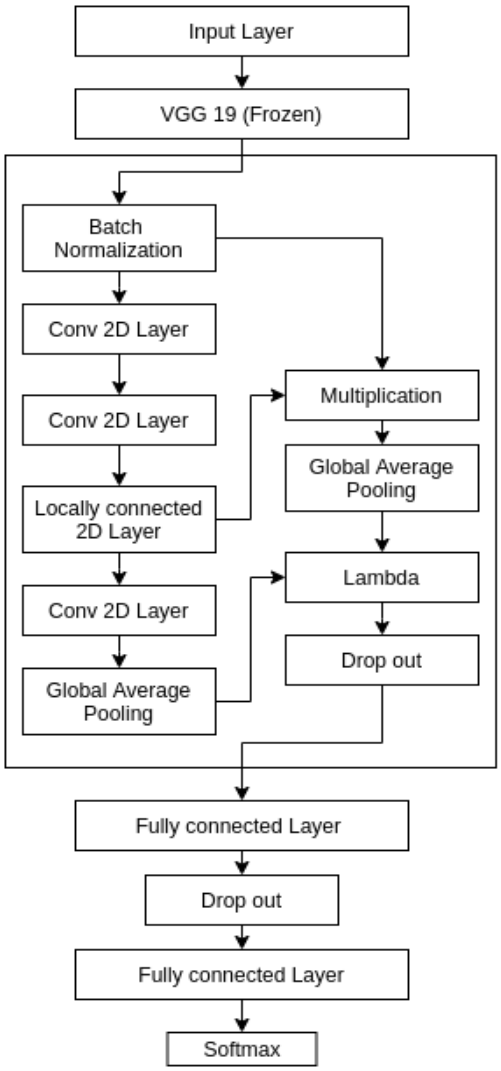


Figure 8. Proposed model architecture along with the attention generation mechanism.

There is problem of class imbalanced clearly shown in Figure 4 in between the classes. There are various techniques of class balancing random oversampling, random undersampling, hybrid sampling and class weighting. We applied class weighting in our models to balance by assigning of score in lower classes. It punishes the errors in the minority classes [64,65]. After class weighting, all classes are equally balanced.

4. Experimental Results

We have proposed a relatively simple architecture that requires next to no preprocessing and can take the benchmark and simple most available dataset of grayscale images. Our methodology requires no special transformation of the dataset, and we do not need to generate any static or dynamic features as a lot of other approaches do. Using a very simple architecture, we can outperform almost all other models. A lot of models that are far more complex and more tedious to train and use have high accuracy scores, but they are severely limited as we have observed in the literature review.

We used Google Colab Cloud for our experiments using Python Language 3.5, Tensorflow, Scikit learn, and the Keras library. The labels were inferred from the dataset by TensorFlow Keras's function called image_dataset_from_directory [66].

In the following subsections, we describe our approach and the framework functionalities that we have used to implement our proposed malware recognition architecture. As we described before in the introduction section. Much of the architecture used to date have been implemented with special processing of input. In this paper, we aim to replace the tedious approaches with something simpler such as attention.

4.1. Experimental Setup

The dataset was split in the most well-used ratio of 70:30 (70% training and 30% testing sets). In total, the training set consists of 8404 files belonging to 25 classes, and the testing set had 935 files belonging to 25 classes. The shape for images for training was (8404, 224, 224, 3) and the shape of labels was (8404, 25).

The attention generation process, which is using 2D Convolutional layers, has a ReLU activation function with the same padding and 2D kernels of size one. The locally connected layer uses sigmoid activation. The upscaling layer that spreads the attention to the other layers simply uses linear activation. The first dense layer uses the exponential linear unit (ELU) activation function, and the last top layer uses SoftMax activation. The model was compiled using the ADAM optimizer with categorical cross-entropy as a loss function, and the metric for performance is accuracy.

The complexity of the proposed models is evaluated using the number of trainable parameters as a proxy. Between both the top and bottom part of the network, the total number of parameters is 20,199,402 which includes the VGG19 weights, the trainable weights in the top layers of CNN, and the fully connected layers. Since we froze the VGG19 layers, the number of trainable parameters was reduced to only 173,482. The number of non-trainable parameters is 20,025,920 as shown in the Table 1.

Table 1. Hyper parameters and other characteristics of proposed models.

Parameter	Attention CNN with VGG19
Batch Size	263
Dropout Rate	0.5
Epochs	10
Learning Rate	0.1 to 0.001
Trainable parameters	173,482
Non-trainable parameters	20,025,920
Loss Function	Categorical Cross entropy
Optimizer	Adam

To construct the best possible model with excellent generalizability and to avoid overfitting the model has been trained using callbacks from Keras we explain them with their parameters here. The most important callbacks we used are early stopping and reduce LR on plateau. The early stopping callback is used to automatically observe the validation loss of a model and according to its parameterization, it decides when to stop the training process. The important parameter is the patience which was set to five after a few experiments. The patience value five means that it will wait for five epochs for the validation loss to decrease by the parameter min delta. By setting the min delta to a low value such as 1×10^{-3}, we wait for a change, mainly a decrease in validation loss of a very small magnitude. This callback is also responsible for restoring the learnt weights from the best epoch as the final weights of the model.

Another important callback was the learning rate scheduler. Since the learning rate can cause problems, it makes sense to change it as the number of epochs grows and the validation loss becomes negligible. The learning rate scheduler used here is called reduce LR on plateau. The important parameters are patience, cool down, and factor. This

works by taking a value of the learning rate that it will continue to reduce by the factor of its parameter factor with the present learning rate. The model will keep observing the validation loss. If the validation loss stops to decrease it will wait for the patience number of epoch and then reduce the learning rate to an even smaller value such as 0.01 or 0.001. This continues to happen until this parameter is no longer able to improve the validation loss. The number of epochs that we start this model to train on is 100 epochs. However, in all of the experiments that we have conducted, we never had to train a model for more than 10–15 epochs. We have obtained our threshold accuracies on epochs as low as four to five. This again shows the high potential of our proposed method.

4.2. Evaluation

We evaluated our result through accuracy, precision, recall, F1- measure, and RUC-AUC curve. We evaluated our results through testing, not only on the malware classes but on benign class as well. It attests to the fact that our model can classify both types of files reasonably. For this experimentation, we have added benign class as an extra class in our dataset. Now we have around three thousand images of benign code binaries in our dataset as well. The results are measured in our models spatial attention CNN with VGG19 class balancing, without class balancing, and with class balancing on benign class as shown in Table 2.

Table 2. Experiment result of our three approaches on SACNN architecture.

Model	Accuracy	Precision	Recall	F1 Score	Specificity	AUC
SACNN VGG19 without class balancing	97.62%	97.68%	97.5%	97.20%	97.68%	97.66%
SACNN VGG19 with class balance	97.42%	97.11%	96.95%	97%	97.33%	97.32%
SACNN VGG19 with class balance on Benign class	97.38%	97.28%	97.56%	97.21%	97.52%	97.45%

Resultantly we have 25 malware classes and 1 benign class in our dataset, making it a 26-class classification problem. We conducted several experiments for this. For this specific experimental setup for the benign class, we have've run multiple experiments and we have obtained a precision of 100%, recall 100% and f1 score of 100%. The AUC of this specific class is also 100%. Table 3 shows the result of extra benign class with 100%.

Table 3. Results of model with an extra benign class.

Model	Accuracy	Precision	Recall	F1-Score
New Benign class SACNN class balancing	100%	100%	100%	100%

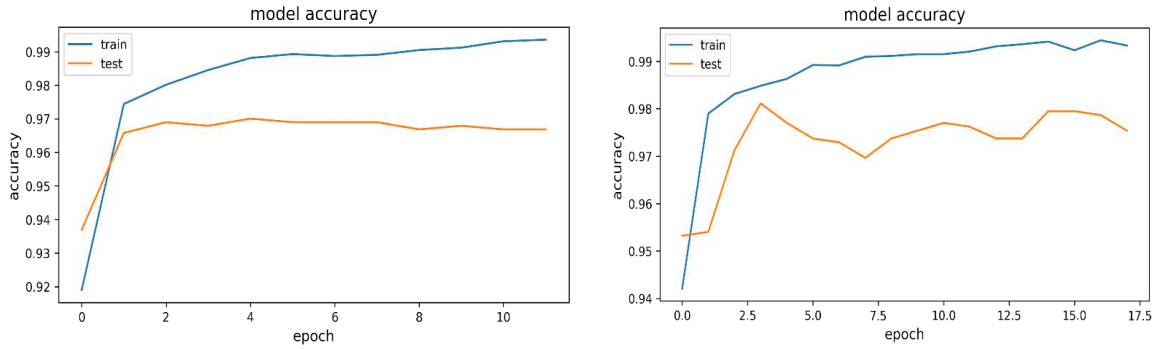
For our model, we have given the average accuracies for models using VGG19. The average accuracy for SACNN with VGG19 with class balancing is higher on fewer epochs of training. The training and validation accuracy are given below in Figure 9a for with class balancing SACNN VGG19 on 25 malware classes, and Figure 9b for with class balancing SACNN VGG19 on benign class.

The models perform very well in both training and testing. The best loss is reached around the fifth epoch and is restored using model callback early stopping which has been described earlier.

Secondly, we also calculated loss value in our proposed. The training and validation loss are given below in Figure 10a for with class balancing SACNN VGG19 on 25 malware classes and Figure 10b for with class balancing SACNN VGG19 on benign class.

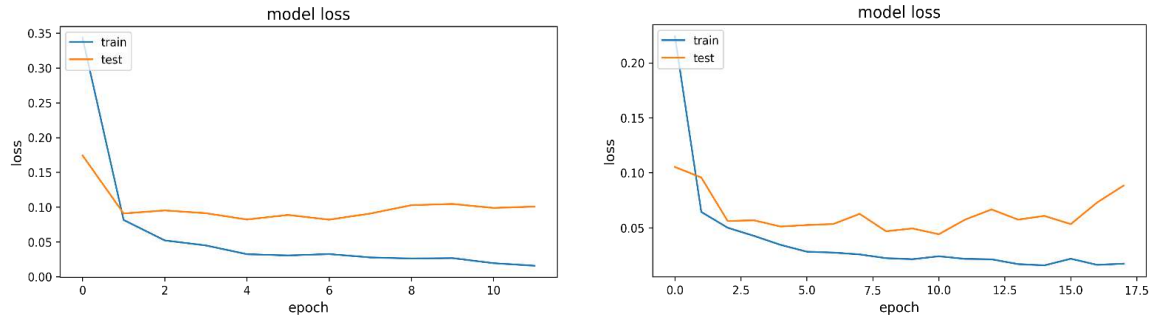
Figure 11 shows the normalized confusion matrix obtained after performing the classification of malware samples into 25 malware classes.

Figure 12 shows the normalized confusion matrix for classification with class balancing and benign class.



(**a**) (**b**)

Figure 9. Training and testing accuracies for spatial attention and convolutional neural network with VGG19 model: (**a**) with class balancing on 25 Malware classes; (**b**) with class balancing on benign class.



(**a**) (**b**)

Figure 10. Training and testing loss for SACNN with VGG19 model: (**a**) with class balancing on 25 Malware classes; (**b**) with class balancing on benign class.

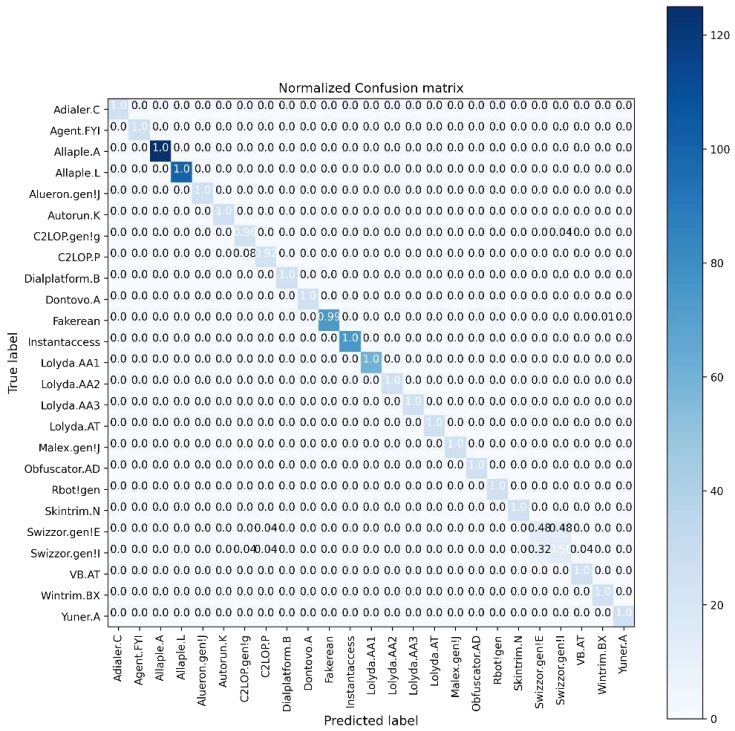


Figure 11. Normalized confusion matrix for classification of 25 malware classes.

The Receiver Operating Characteristic (ROC) curves with calculated area under curve (AUC) values of classification into 25 malware classes are shown in Figure 13.

Figure 14 shows ROC curves with AUC values for classification with class balancing and benign class.

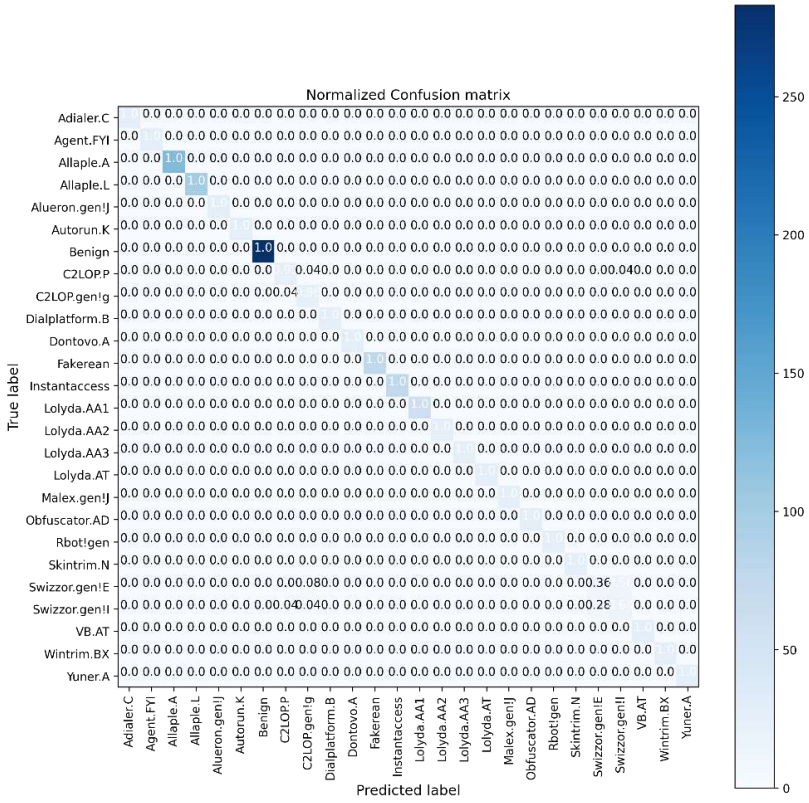


Figure 12. Normalized confusion matrix for classification with class balancing and benign class.

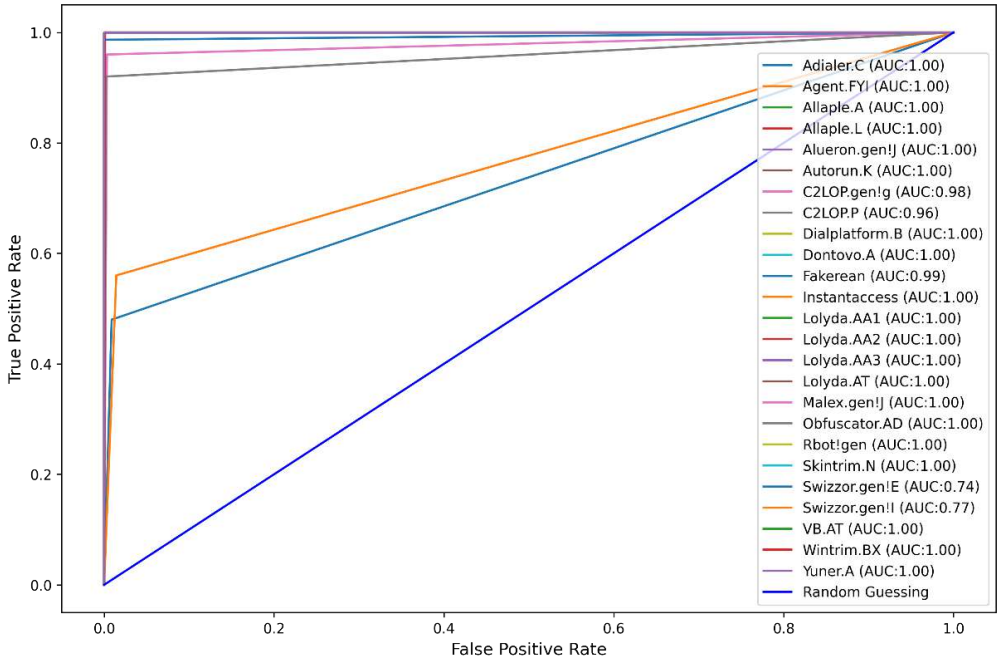


Figure 13. ROC (receiver operating characteristic) curves with area under curve values for classification into 25 malware families.

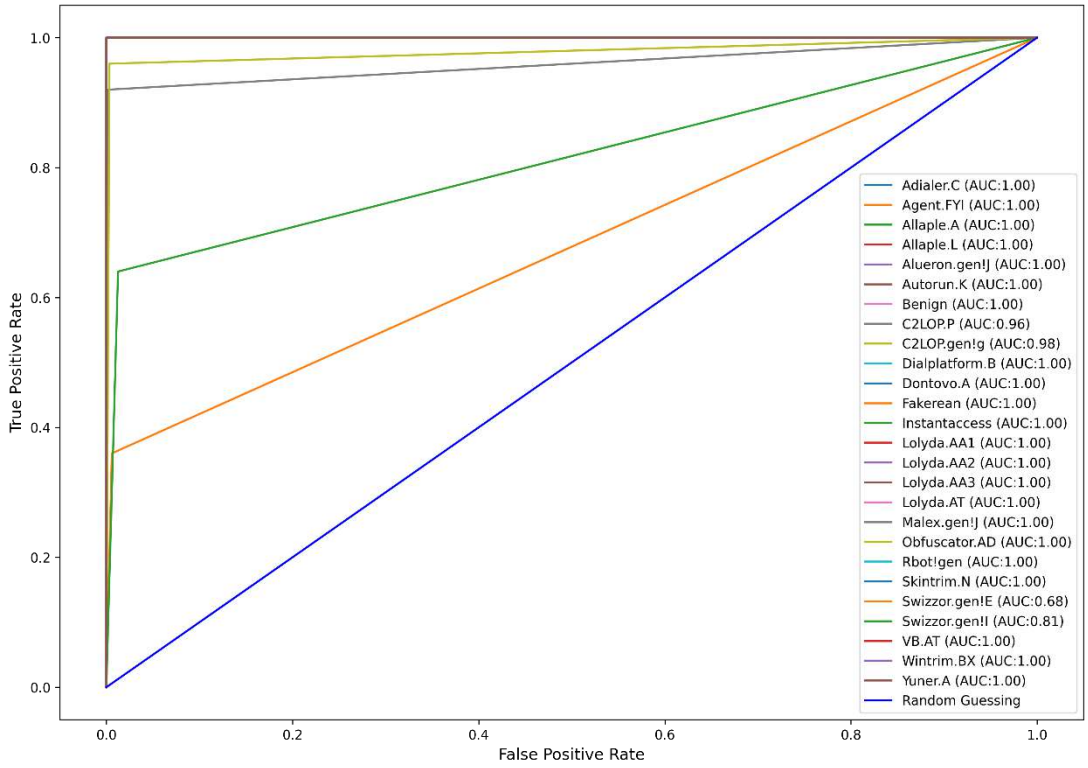


Figure 14. ROC curves with area under curve values for classification with class balancing and benign class.

5. Discussion

The spatial attention-based deep learning model proposed in this paper has some advantages over other machine learning and deep learning models proposed by other authors for the malware recognition task. Most of the models use some type of feature engineering techniques and require specialized algorithms, while other models just too heavily dependent on the feature generation and image transformations with data augmentation. All of this requires too much of human intervention to specifically pre-process the data and/or create or select the malware features. These types of pipelines have become useless since the speed of new malware generation has grown exponentially. In face of all this complexity, we have proposed a simple model that produces better results with next to no complexity in its working. It used spatial attention generated using CNNs and using feedforward and dropout layers with lesser number of trainable parameters, whilst it can outperform most of the other deep learning models for malware recognition. We also compared our result with previous studies as shown in Table 4.

To compare our method with state-of-the-art models we first need to understand that most state-of-the-art models utilize many types of data pre-processing for image augmentation in order to produce good results. Our approach utilizes most basic self-learning capabilities of neural networks and combines them with a simple spatial attention mechanism to avoid this pre-processing, which results in a lightweight model considering the operations it performs for attention generation. Multi-head attention-based models such as transformers have a wide application in computer vision, but the intricate and extensive approach is not what we are aiming. Our model is not light weight, since it holds millions of parameters from the convolution based VGG16 base model alone and our top model only adds more weights to the system. As shown in the above table, our claims hold true since our model is outperforming system which use state of the art techniques.

Table 4. Comparison of the proposed model with state-of-the-art models.

Architecture	Accuracy	Year & Reference
GoogleNET	84.0%	(2018) [22]
ResNET (Avg)	86.01%	
CNN-SVM	77.22%	
GRU-SVM	84.92%	(2019) [26]
MLP-SVM	80.46%	
VGG16 + SVM	92.97%	(2018) [20]
Linear-binary patterns and CNN (LBP + CNN)	93.17%	(2017) [62]
Custom Deep Learning architecture + KNN (fast.ai)	94.80%	(2019) [21]
SVM with T + C + L features	95.23%	(2018) [24]
KNN with T + C + L features	96.23%	
CNN-2 layer-LSTM (7:3)	96.4%	(2019) [27]
CNN-2 layer-LSTM (9:1)	96.8%	
SSPNet1 (with spatial pyramid pooling)	96.60%	(2020) [63]
Multi-Objective learning	96.86%	(2019) [34]
Capsule network model CapsNet	96.58%	(2021) [44]
Kernel-based ELM with statistical texture features	94.25%	(2020) [43]
Spatial Attention CNN with VGG16 without Class Balance Malware	97.62%	Our study
Spatial Attention CNN with VGG16 with Class Balance Malware	97.42%	
Spatial Attention CNN with VGG19 with Class Balance Benign	97.38%	

Our results we have presented here have a few limitations as well, the first and biggest of which arise from the dataset used for performance evaluation itself. The dataset has a huge imbalance in the number of examples per malware class, while some classes have more than two thousand examples whereas others have less than a hundred due to which the malware classes with the lower number of examples unfortunately have higher misclassification rates. The other limitation of this work is the lack of the exploration in the data augmentation and the feature engineering domains.

In future we could use image augmentation to improve malware recognition result. As we used spatial attention which is a recently developed concept. Lately more SOTA models and methods are being developed on this topic which we haven't explored yet. In future we hope to compare and enhance our methods with the use of vision transformers and more types of visual attention mechanisms without convolutional pipeline [67,68].

Recently big data using Spark ML and big deep learning (BigDL) was performed very good result due to Spark framework so we can apply our work through big data in IOT malware prediction [69–75].

6. Conclusions

In this paper we have proposed to use attention enhancement via CNNs to solve the malware recognition problem without the need of a feature engineering technique or hand-made feature design. We have aimed to show how the attention mechanism with CNNs can be used and how this type of spatial attention, which demonstrated enormous advantages in computer vision problems, can be applied in images-based malware detection as well.

We have proposed a novel approach based on convolutional neural networks (CNN) that uses spatial convolutional attention to classify malware into 25 different malware families. The performance was evaluated on the Malimg benchmark dataset, achieving an accuracy of 97.68%. The experimental results indicate our proposed model can be used for image-based malware detection with the state-of-the-art results despite being simpler as compared to other solutions. Attention itself is a wide topic, and so many variants of attention have emerged in recent years. This provides an opportunity to further enhance this solution for use in smart homes and other Internet-of-Things (IoT) environments. Using our proposed attention-based model, and improving upon it using various deep learning techniques, it is possible to develop a generic pipeline that uses deep learning models enhanced by attention to solving malware recognition problems more effectively.

Author Contributions: Conceptualization, M.J.A.; Data curation, O.A.M.; Formal analysis, O.A.M., A.Y., A.M.Z., R.D. and K.H.A.; Funding acquisition, R.D.; Investigation, O.A.M. and K.H.A.; Methodology, M.J.A.; Project administration, M.A.M.; Resources, A.M.Z.; Software, O.A.M.; Supervision, M.A.M.; Validation, A.Y., A.M.Z. and K.H.A.; Writing–original draft, M.J.A., O.A.M. and A.Y.; Writing–review & editing, M.A.M. and R.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The Malimg dataset [33] used in this study is openly available from: https://www.dropbox.com/s/ep8qjakfwh1rzk4/malimg_dataset.zip?dl=0, accessed on 20 June 2011.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rieck, K.; Trinius, P.; Willems, C.; Holz, T. Automatic analysis of malware behavior using machine learning. *J. Comput. Secur.* **2011**, *19*, 639–668. [CrossRef]
2. Awan, M.J.; Farooq, U.; Babar, H.M.A.; Yasin, A.; Nobanee, H.; Hussain, M.; Hakeem, O.; Zain, A.M. Real-time DDoS attack detection system using big data approach. *Sustainability* **2021**, *13*, 10743. [CrossRef]
3. Ferooz, F.; Hassan, M.T.; Awan, M.J.; Nobanee, H.; Kamal, M.; Yasin, A.; Zain, A.M. Suicide bomb attack identification and analytics through data mining techniques. *Electronics* **2021**, *10*, 2398. [CrossRef]
4. Belbus, N.V.; Yeo, S.-S.; Cho, E.-S.; Kim, J.-A. Malware and antivirus deployment for enterprise IT security. In Proceedings of the 2008 International Symposium on Ubiquitous Multimedia Computing, Hobart, Australia, 13–15 October 2008.
5. Azeez, N.A.; Salaudeen, B.B.; Misra, S.; Damasevicius, R.; Maskeliunas, R. Identifying phishing attacks in communication networks using URL consistency features. *Int. J. Electron. Secur. Digit. Forensics* **2021**, *12*, 200–213. [CrossRef]
6. Yong, B.; Wei, W.; Li, K.; Shen, J.; Zhou, Q.; Wozniak, M.; Polap, D.; Damaševičius, R. Ensemble machine learning approaches for webshell detection in internet of things environments. *Trans. Emerg. Telecommun. Technol.* **2020**. [CrossRef]
7. Mohammed, M.A.; Ibrahim, D.A.; Salman, A.O. Adaptive intelligent learning approach based on visual anti-spam email model for multi-natural language. *J. Intell. Syst.* **2021**, *30*, 774–792. [CrossRef]
8. Rehma, A.A.; Awan, M.J.; Butt, I. Comparison and evaluation of information retrieval models. *VFAST Trans. Softw.* **2018**, *6*, 7–14.
9. Rhode, M.; Burnap, P.; Jones, K. Early-stage malware prediction using recurrent neural networks. *Comput. Secur.* **2018**, *77*, 578–594. [CrossRef]
10. Alam, T.M.; Awan, M.J. Domain analysis of information extraction techniques. *Int. J. Multidiscip. Sci. Eng.* **2018**, *9*, 1–9.
11. Adebayo, O.S.; Abdul Aziz, N. Improved malware detection model with apriori association rule and particle swarm optimization. *Secur. Commun. Netw.* **2019**, *2019*, 1–13. [CrossRef]
12. Ali, Y.; Farooq, A.; Alam, T.M.; Farooq, M.S.; Awan, M.J.; Baig, T.I. Detection of schistosomiasis factors using association rule mining. *IEEE Access* **2019**, *7*, 186108–186114. [CrossRef]
13. Akram, R.N.; Chen, H.; Lopez, J.; Sauveron, D.; Yang, L.T. Security, privacy and trust of user-centric solutions. *Future Gener. Comput. Syst.* **2018**, *80*, 417–420. [CrossRef]
14. Anderson, H.S.; Kharkar, A.; Filar, B.; Roth, P. Evading machine learning malware detection. In Proceedings of the Black Hat, Las Vegas, NV, USA, 22–27 July 2017; pp. 1–6.
15. Azeez, N.A.; Odufuwa, O.E.; Misra, S.; Oluranti, J.; Damaševičius, R. Windows PE malware detection using ensemble learning. *Informatics* **2021**, *8*, 10. [CrossRef]
16. Khalaf, B.A.; Mostafa, S.A.; Mustapha, A.; Mohammed, M.A.; Mahmoud, M.A.; Al-Rimy, B.A.S.; Abd Razak, S.; Elhoseny, M.; Marks, A. An adaptive protection of flooding attacks model for complex network environments. *Secur. Commun. Netw.* **2021**, 2021. [CrossRef]
17. Anam, M.; Ponnusamy, V.A.; Hussain, M.; Nadeem, M.W.; Javed, M.; Gou, H.G.; Qadeer, S. Osteoporosis prediction for trabecular bone using machine learning: A review. *Comput. Mater. Contin.* **2021**, *67*, 89–105. [CrossRef]
18. Azizan, A.H.; Mostafa, S.A.; Mustapha, A.; Foozy, C.F.M.; Abd Wahab, M.H.; Mohammed, M.A.; Khalaf, B.A. A machine learning approach for improving the performance of network intrusion detection systems. *Ann. Emerg. Technol. Comput. (AETiC)* **2021**, *5*, 201–208. [CrossRef]
19. Gupta, M.; Jain, R.; Arora, S.; Gupta, A.; Awan, M.J.; Chaudhary, G.; Nobanee, H. AI-enabled COVID-19 outbreak analysis and prediction: Indian states vs. union territories. *Comput. Mater. Contin.* **2021**, *67*, 933–950. [CrossRef]
20. Damaševičius, R.; Venčkauskas, A.; Toldinas, J.; Grigaliūnas, Š. Ensemble-based classification using neural networks and machine learning models for windows pe malware detection. *Electronics* **2021**, *10*, 485. [CrossRef]
21. Awan, M.J.; Yasin, A.; Nobanee, H.; Ali, A.A.; Shahzad, Z.; Nabeel, M.; Zain, A.M.; Shahzad, H.M.F. Fake news data exploration and analytics. *Electronics* **2021**, *10*, 2326. [CrossRef]
22. Lal, S.; Rehman, S.U.; Shah, J.H.; Meraj, T.; Rauf, H.T.; Damaševičius, R.; Mohammed, M.A.; Abdulkareem, K.H. Adversarial attack and defence through adversarial training and feature fusion for diabetic retinopathy recognition. *Sensors* **2021**, *21*, 3922. [CrossRef]

23. Alharbi, A.; Alosaimi, W.; Alyami, H.; Rauf, H.T.; Damaševičius, R. Botnet attack detection using local global best bat algorithm for industrial internet of things. *Electronics* **2021**, *10*, 1341. [CrossRef]
24. Mahdavifar, S.; Ghorbani, A.A. Application of deep learning to cybersecurity: A survey. *Neurocomputing* **2019**, *347*, 149–176. [CrossRef]
25. Conti, G.; Dean, E.; Sinda, M.; Sangster, B. Visual reverse engineering of binary and data files. In Proceedings of the International Workshop on Visualization for Computer Security, Cambridge, MA, USA, 15 September 2008; pp. 1–17.
26. Nagi, A.T.; Awan, M.J.; Javed, R.; Ayesha, N. A Comparison of two-stage classifier algorithm with ensemble techniques on detection of diabetic retinopathy. In Proceedings of the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), Riyadh, Saudi Arabia, 6–7 April 2021; pp. 212–215.
27. Abdullah, A.; Awan, M.; Shehzad, M.; Ashraf, M. Fake news classification bimodal using convolutional neural network and long short-term memory. *Int. J. Emerg. Technol. Learn* **2020**, *11*, 209–212.
28. Mujahid, A.; Awan, M.J.; Yasin, A.; Mohammed, M.A.; Damaševičius, R.; Maskeliūnas, R.; Abdulkareem, K.H. Real-time hand gesture recognition based on deep learning YOLOv3 Model. *Appl. Sci.* **2021**, *11*, 4164. [CrossRef]
29. Rezende, E.; Ruppert, G.; Carvalho, T.; Ramos, F.; De Geus, P. Malicious software classification using transfer learning of resnet-50 deep neural network. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 1011–1014.
30. Khan, R.U.; Zhang, X.; Kumar, R. Analysis of ResNet and GoogleNet models for malware detection. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 29–37. [CrossRef]
31. 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. [CrossRef]
32. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? *arXiv* **2014**, arXiv:1411.1792.
33. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburgh, PA, USA, 20 June 2011; pp. 1–7.
34. Agarap, A.F. Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification. *arXiv* **2017**, arXiv:1801.00318 2017.
35. Akarsh, S.; Poornachandran, P.; Menon, V.K.; Soman, K. A Detailed investigation and analysis of deep learning architectures and visualization techniques for malware family identification. In *Cybersecurity and Secure Information Systems*; Springer: New York, NY, USA, 2019; pp. 241–286.
36. Akarsh, S.; Simran, K.; Poornachandran, P.; Menon, V.K.; Soman, K. Deep learning framework and visualization for malware classification. In Proceedings of the 2019 5th International Conference on Advanced Computing Communication Systems (ICACCS), Coimbatore, India, 15–16 March 2019; pp. 1059–1063.
37. Kumar, S. MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in internet of things. *Future Gener. Comput. Syst.* **2021**, *125*, 334–351.
38. Vinayakumar, R.; Alazab, M.; Soman, K.; Poornachandran, P.; Venkatraman, S. Robust intelligent malware detection using deep learning. *IEEE Access* **2019**, *7*, 46717–46738. [CrossRef]
39. Xiao, G.; Li, J.; Chen, Y.; Li, K. MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks. *J. Parallel Distrib. Comput.* **2020**, *141*, 49–58. [CrossRef]
40. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.-G.; Chen, J. Detection of malicious code variants based on deep learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]
41. Cui, Z.; Du, L.; Wang, P.; Cai, X.; Zhang, W. Malicious code detection based on CNNs and multi-objective algorithm. *J. Parallel Distrib. Comput.* **2019**, *129*, 50–58. [CrossRef]
42. Jain, M.; Andreopoulos, W.; Stamp, M. CNN vs ELM for image-based malware classification. *arXiv* **2021**, arXiv:2103.13820.
43. Naeem, H.; Ullah, F.; Naeem, M.R.; Khalid, S.; Vasan, D.; Jabbar, S.; Saeed, S. Malware detection in industrial internet of things based on hybrid image visualization and deep learning model. *Ad Hoc Networks* **2020**, *105*, 102154. [CrossRef]
44. Venkatraman, S.; Alazab, M.; Vinayakumar, R. A hybrid deep learning image-based analysis for effective malware detection. *J. Inf. Secur. Appl.* **2019**, *47*, 377–389. [CrossRef]
45. Vu, D.-L.; Nguyen, T.-K.; Nguyen, T.V.; Nguyen, T.N.; Massacci, F.; Phung, P.H. A convolutional transformation network for malware classification. In Proceedings of the 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), Hanoi, Vietnam, 12–13 December 2019; pp. 234–239.
46. El-Shafai, W.; Almomani, I.; Alkhayer, A. Visualized malware multi-classification framework using fine-tuned CNN-based transfer learning models. *Appl. Sci.* **2021**, *11*, 6446. [CrossRef]
47. Moussas, V.; Andreatos, A. Malware detection based on code visualization and two-level classification. *Information* **2021**, *1*, 118. [CrossRef]
48. Roseline, S.A.; Geetha, S.; Kadry, S.; Nam, Y. Intelligent vision-based malware detection and classification using deep random forest paradigm. *IEEE Access* **2020**, *8*, 206303–206324. [CrossRef]
49. Verma, V.; Muttoo, S.K.; Singh, V.B. Multiclass malware classification via first-and second-order texture statistics. *Comput. Secur.* **2020**, *97*, 101895. [CrossRef]
50. Çayır, A.; Ünal, U.; Dağ, H. Random CapsNet forest model for imbalanced malware type classification task. *Comput. Secur.* **2021**, *102*, 102133. [CrossRef]

51. Wozniak, M.; Silka, J.; Wieczorek, M.; Alrashoud, M. Recurrent neural network model for IoT and networking malware threat detection. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5583–5594. [CrossRef]
52. Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševičius, R.; Blažauskas, T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. *Appl. Sci.* **2020**, *10*, 4966. [CrossRef]
53. Hemalatha, J.; Roseline, S.A.; Geetha, S.; Kadry, S.; Damaševičius, R. An efficient densenet-based deep learning model for malware detection. *Entropy* **2021**, *23*, 344. [CrossRef] [PubMed]
54. Toldinas, J.; Venčkauskas, A.; Damaševičius, R.; Grigaliūnas, Š.; Morkevičius, N.; Baranauskas, E. A novel approach for network intrusion detection using multistage deep learning image recognition. *Electronics* **2021**, *10*, 1854. [CrossRef]
55. Wang, C.; Zhao, Z.; Wang, F.; Li, Q. A novel malware detection and family classification scheme for IoT based on DEAM and DenseNet. *Secur. Commun. Netw.* **2021**, *2021*, 1–16.
56. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
57. Awan, M.J.; Raza, A.; Yasin, A.; Shehzad, H.M.F.; Butt, I. The customized convolutional neural network of face emotion expression classification. *Ann. Rom. Soc. Cell Biol.* **2021**, *25*, 5296–5304.
58. Mubashar, R.; Awan, M.J.; Ahsan, M.; Yasin, A.; Singh, V.P. Efficient residential load forecasting using deep learning approach. *Int. J. Comput. Appl. Technol.* **2021**.
59. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
60. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
61. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
62. Awan, M.J.; Bilal, M.H.; Yasin, A.; Nobanee, H.; Khan, N.S.; Zain, A.M. Detection of COVID-19 in chest X-ray images: A big data enabled deep learning approach. *Int. J. Environ. Res. Public Health* **2021**, *18*, 10147. [CrossRef]
63. Ding, E.; Cheng, Y.; Xiao, C.; Liu, Z.; Yu, W. Efficient attention mechanism for dynamic convolution in lightweight neural network. *Appl. Sci.* **2021**, *11*, 3111. [CrossRef]
64. Javed Awan, M.; Mohd Rahim, M.S.; Salim, N.; Mohammed, M.A.; Garcia-Zapirain, B.; Abdulkareem, K.H. Efficient detection of knee anterior cruciate ligament from magnetic resonance imaging using deep learning approach. *Diagnostics* **2021**, *11*, 105. [CrossRef] [PubMed]
65. Johnson, J.M.; Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J. Big Data* **2019**, *6*, 27. [CrossRef]
66. Awan, M.J.; Rahim, M.S.M.; Salim, N.; Ismail, A.W.; Shabbir, H. Acceleration of knee MRI cancellous bone classification on google colaboratory using convolutional neural network. *Int. J. Adv. Trends Comput. Sci.* **2019**, *8*, 83–88. [CrossRef]
67. Yang, J.; Li, C.; Zhang, P.; Dai, X.; Xiao, B.; Yuan, L.; Gao, J. Focal self-attention for local-global interactions in vision transformers. *arXiv* **2021**, arXiv:2107.00641.
68. Tran, V.-N.; Lee, S.-H.; Le, H.-S.; Kwon, K.-R. High Performance deepfake video detection on CNN-based with attention target-specific regions and manual distillation extraction. *Appl. Sci.* **2021**, *11*, 7678. [CrossRef]
69. Awan, M.J.; Khan, R.A.; Nobanee, H.; Yasin, A.; Anwar, S.M.; Naseem, U.; Singh, V.P. A Recommendation engine for predicting movie ratings using a big data approach. *Electronics* **2021**, *10*, 1215. [CrossRef]
70. Awan, M.J.; Rahim, M.S.M.; Nobanee, H.; Munawar, A.; Yasin, A.; Zain, A.M. Social media and stock market prediction: A big data approach. *Comput. Mater. Contin.* **2021**, *67*, 2569–2583. [CrossRef]
71. Ahmed, H.M.; Awan, M.J.; Khan, N.S.; Yasin, A.; Shehzad, H.M.F. Sentiment analysis of online food reviews using big data analytics. *Elem. Educ. Online* **2021**, *20*, 827–836.
72. Aftab, M.O.; Awan, M.J.; Khalid, S.; Javed, R.; Shabir, H. Executing spark BigDL for leukemia detection from microscopic images using transfer learning. In Proceedings of the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), Riyadh, Saudi Arabia, 6–7 April 2021; pp. 216–220.
73. Awan, M.J.; Khan, M.A.; Ansari, Z.K.; Yasin, A.; Shehzad, H.M.F. Fake profile recognition using big data analytics in social media platforms. *Int. J. Comput. Appl. Technol.* **2021**. [CrossRef]
74. Awan, M.J.; Rahim, M.S.M.; Nobanee, H.; Yasin, A.; Khalaf, O.I.; Ishfaq, U. A big data approach to black friday sales. *Intell. Autom. Soft Comput.* **2021**, *27*, 785–797. [CrossRef]
75. Awan, M.J.; Gilani, S.A.H.; Ramzan, H.; Nobanee, H.; Yasin, A.; Zain, A.M.; Javed, R. Cricket match analytics using the big data approach. *Electronics* **2021**, *10*, 2350. [CrossRef]