AJEMA GLOBAL RESEARCH NETWORK

AMERICAN Journal of Engineering, Mechanics and Architecture
Volume 2, Issue 10, 2024 ISSN (E): 2993-2637

Abnormal Files Classification using Convolutional Neural Network CNN

Mohammad Musaddak
Al-Farabi University, Iraq, Baghdad

Abstract: As of late, turning Windows files into pictures and analyzing them with deep learning and machine learning have been regarded as state-of-the-art methods for identifying and classifying malware. This is mostly because deep learning model performance in image classification has recently experienced a boom in success, and image-based malware detection and classification is platform independent. Convolutional neural network (CNN) deep learning techniques are successfully used for image-based Windows malware classification, according to a review of the literature. Nevertheless, just a small percentage of the entire picture representation had the infection. Finding and identifying these impacted little areas is crucial to achieving a high degree of malware classification accuracy. This study locates and identifies the little contaminated spots in the overall picture by integrating a Data augmentation technique with a CNN. On a dataset of malware images, a thorough examination and analysis of the suggested technique were conducted. The effectiveness of the suggested Data augmentation-based CNN method was evaluated against many non-attention-CNN-based methods using different data splits from the benchmark malware picture training and testing dataset. The proposed CNN approach ensured computational efficiency and outperformed non-attention-based CNN methods in all the data-splits. Most notably, the majority of the techniques demonstrate consistent performance across all training and testing data splits, illuminating CNN's multi-headed attention. model's generalizability to perform on the diverse datasets.

Introduction

The majority of anti-virus products on the market today are signature-based, having gained popularity in the early days of malware detection. Raw binaries were examined and a signature based on n-grams was created for signature-based malware detection [1]. This kind of strategy is straightforward and precise. Nevertheless, in order to handle newly discovered malware or variations of already-existing malware, the signature database must be updated on a regular basis. The typical methodologies for malware analysis that were most often used were heuristic-based techniques, hybrid analysis, static analysis, and dynamic analysis [2]. .All of these techniques, taken together, entail scanning systems to find harmful or unauthorized activity. The suspicious files are then examined by the security administrator, and either a quarantine is placed on them or an update is applied to the susceptible system. Every strategy frequently reacts slowly to fresh dangers and attacks. By examining the computer codes and creating control flow graphs without running the program, a method known as static analysis is used. Nevertheless, code obfuscation is outside the scope of static analysis. With dynamic analysis, the malware is run in a sandbox, simulator, or virtual environment, and its execution track is examined (behavior analysis). While it is a promising method for handling code obfuscation, it is difficult, time-consuming, and computationally costly. The malwares were first converted into grayscale

images, and [4] used the GIST feature in conjunction with the k-nearest neighbor classifier to classify the trojan. The Malimg dataset and the author's thorough examination and analysis were made freely available for study. Owing to the current explosion in deep learning [7, 8], primarily to CNN's success, numerous techniques have been put out and assessed by researchers for the Malimg dataset's use in malware classification. Furthermore, for a variety of applications, deep learning-based approaches outperformed the other techniques [9], [10], [11], [12]. The comprehensive review of the literature found in Section 2. For malware detection and classification, the majority of currently used techniques rely on CNN or feature extraction with machine learning classifiers. These days, malware detection and classification using deep learning-based techniques that turn malware into images work well. This is primarily because these techniques fully steer clear of manual feature engineering techniques. Because manual feature engineering techniques necessitate domain expertise and perform poorly when applied to complicated and unknown data. Nevertheless, these techniques fail to recognize even a small portion of the virus image's affected areas. The majority of the time, virus writers merely slightly alter the original files. By applying filters to a picture, CNN can improve its feature representation; but, its ability to recognize afflicted areas is severely restricted. Security experts increased the breadth and depth of the CNN network in order to enhance CNN's effectiveness on malware image-based tasks [13, 14].

Literature Survey

An technique to malware detection known as custom-trained transfer learning was introduced by Marastoni et al. [34]. They trained the CNN model as a transfer-learned model using a bespoke OBF dataset that was produced utilizing obfuscation techniques. The MalImg and MsM2015 datasets were utilized to train an additional CNN and LSTM model, which was then used to predict each other's malware dataset and confirm prediction accuracy. They attained LSTM-level accuracy and suggested using bicubic interpolation to obtain consistent image size. But without data balance, transfer learning showed mediocre performance; moreover, the models required fixed-size pictures, which prolonged the time necessary for data pre-processing; and the obfuscation techniques needed to be improved.

CNN and MLP models should be used for malware sample analysis and detection, according to Kim et al. [36]. They used their method to the publicly accessible Microsoft Malware dataset. Some of the data in the dataset were eliminated and converted into an abstract visual graph in order to determine which category the virus is located in. Through comparison investigation, the degree of similarity between this malware was discovered. It has been discovered that artificial intelligence deep learning is a useful technique for precisely and swiftly identifying malware using images. This technique outperformed the conventional signature-based strategy in identifying recently discovered malware. It was discovered that the MLP classifier needs to be upgraded since it was unreliable.

In order to fully utilize deep learning models, Sharma et al. [40] used a classification model for malware that combined them with CNN and other machine learning classifiers. By using mathematical functions, these models made it possible to identify malware that was just released. In terms of approach, CNN alone did the best on the MalImg dataset. Architecture for the CNN-SVM model required to be upgraded. For multi-class situations, this would entail utilizing many SVMs, increasing the size of the model and hence the calculation time.

In [27 p3], a CNN is trained with a multi-headed attention-based approach to identify and categorize the minuscule afflicted regions within the whole picture. On many data splits of the testing and training malware image benchmark dataset, the anticipated multiheaded attention-based CNN approach performed comparably to other non-attention-CNN-based strategies.

Using the fusion feature set paradigm, Chaganti et al. [28 p3] presented a DL-based CNN technique to identify malware on Portable Executable (PE) dual files. We provided a comprehensive evaluation of the performance of several DL technique structures and machine

learning classifiers, such as Support Vector Machines (SVM), using multi-aspect feature sets that included static, dynamic, and picture characteristics. The created CNN technique.

Opcode sequences were employed in earlier research on word2vec-based long short-term memory (LTSM) [20 p4], and an LSTM was utilized to take into account the opcodes' sequential order in the file. They passed the obtained representation to the LSTM after encoding opcodes using the word2vec embedding technique. In order to classify malware, Qiao et al. suggested transforming grayscale pictures using word vector similarity determined from opcodes and sending the results to a CNN [21 p4].

In [18 p5], an advanced learning machine (ELM) and a CNN were utilized to classify malware. Comparing the two algorithms' performance in identifying malware photos is the primary objective of this paper. The results show that ELMs are better appropriate for the task at hand, being quicker and more accurate than CNNs. This study trains their models using photos from the dataset of Nataraj et al. [30 p5].

The deep residual network architecture with 50 layers from [16 p5] is used by the authors in [40 p5] to categorize malware from the MalImg dataset. In order to categorize the malware samples into their respective families, the network is first trained on a common object identification task. After that, the last layer is removed from the model and a new dense model with 25 output nodes is created. Slight modifications to the previously described method have been suggested by other recent publications, such [9 p5] and [5 p5], who always pre-train their model on typical picture classification tasks.

Convolutional Neural Network

CNNs are popularly employed for image identification and classification. They are feed-forward neural network models that are inspired by the human visual cortex. Their translational invariance has been credited with their performance in image-related learning tasks, enabling them to be applied to issues like handwritten character recognition and face recognition. CNNs perform a great job of extracting spatial information from the data in general. CNNs are fundamentally made up of a dense layer coupled to at least one convolutional layer. During the training phase, the convolutional layer applies a convolution to the input to separate the features that the network considers significant. The input photos are reduced to a smaller feature set by this procedure, which significantly lowers the number of weights required [23 p6]. The max pooling layer, which is often coupled to the convolutional network, helps further minimize the size of the features identified by aggregating the values of several neurons into a single value, which is typically either the average value or the maximum value (hence max-pooling). At least one completely linked (or dense) layer completes the CNN architecture after a combination of the aforementioned layers. This layer operates as a multi-layer perceptron and is in charge of the actual classification process. It receives as input the characteristics that were taken out of the earlier layers [24 p6]. Regularization methods are widely used and may be implemented anywhere in the network to prevent overfitting. Better generalization is made possible as a result of the network being less likely to memorize the training set [22 p6].

CNN consist of :

Convolutional Layer: First, the CNN input format is covered. While CNN employs a multi-channeled image as input, typical neural networks use vector formats. For instance, the RGB image format has three channels, but the grayscale picture format just has one. To further understand the convolutional process, consider a 4 x 4 grayscale image with a 2 x 2 random weight-initialized kernel. The kernel first scans the whole picture in both horizontal and vertical directions. Furthermore, the kernel's dot product and the input image are computed, which involves multiplying and adding each of their distinct values to produce a single scalar result. The procedure is then carried out again until sliding is no longer feasible. The calculated dot product values reflect the output feature map. In Figure 1, the fundamental calculations carried out at every step are displayed clearly. In this illustration, the 2 2 kernel is represented by the

light green hue, while a piece of the same-sized input image is represented by the light blue hue. Both are multiplied, and after adding together the final product values, the result (shown in light orange) represents an input value into the output feature map. In the example above, a stride of one is applied to the kernel (represented by the chosen step-size over all vertical and horizontal places) instead of padding the input picture. Using a different stride value is also possible. Raising the stride value also results in a feature map that is smaller in size.

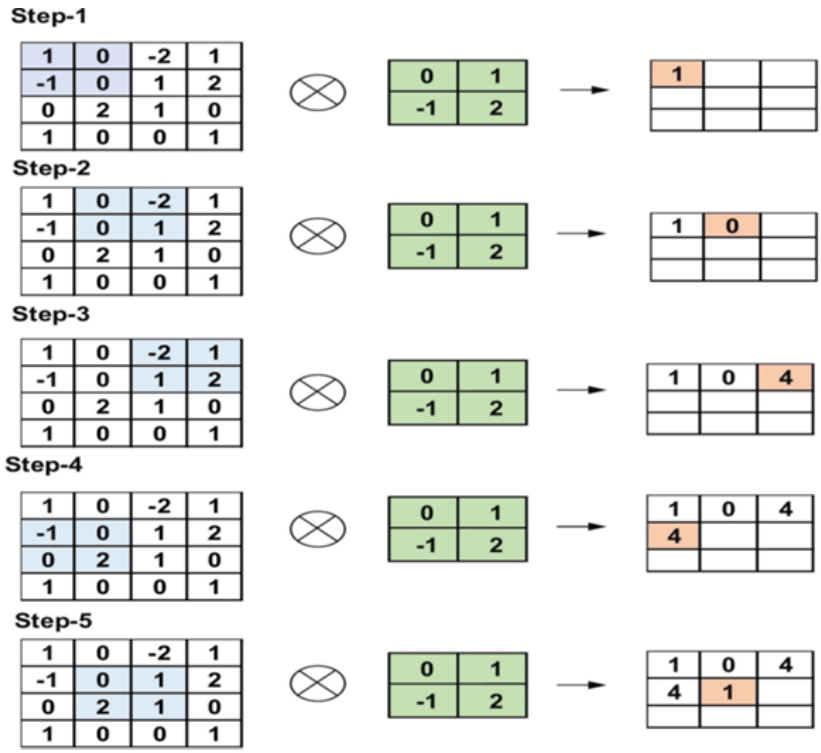

Figure 1 Convolutional Operation

Pooling Layer: Subsampling is the main function of the pooling layer for the feature maps. Convolutional methods are used in the production of these maps. Put otherwise, this method reduces the size of bigger feature maps into smaller ones. It also keeps most of the dominant information (or attributes) during the pooling process. Before the pooling procedure, the size of the kernel and the stride are allocated. Different pooling layers can employ a variety of pooling algorithm types. Global max pooling, min pooling, max pooling, average pooling, and global average pooling (GAP) are a few of these options. The maximum, minimum, and GAP techniques are the most popular and often applied pooling algorithms. These three pooling strategies are shown in Figure 2.

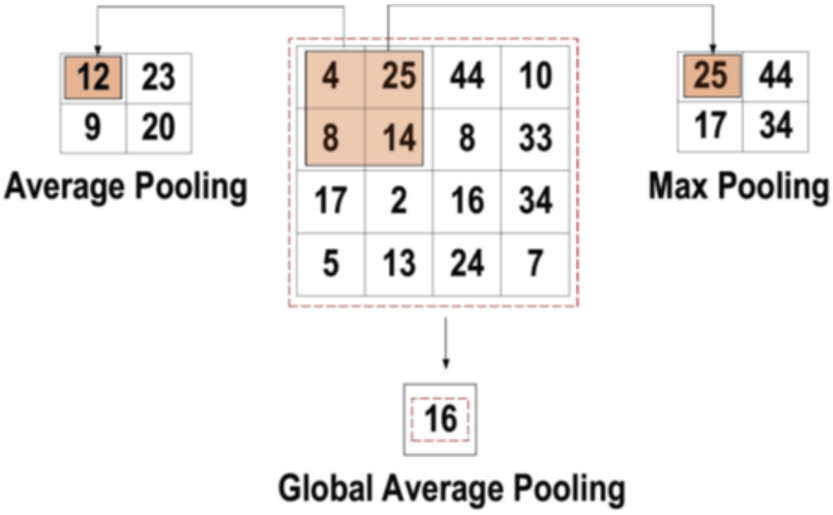

Figure 2 Pooling layer

Fully Connected layer: In most CNN architectures, this layer is found at the end. The Fully Connected (FC) technique connects every cell in this layer to every cell in the layer above. It functions as a classifier for CNN. Because it is a feed-forward ANN, it uses the same fundamental method as a traditional multiple-layer perceptron neural network. The FC layer receives its input from the preceding pooling or convolutional layer. To construct a vector representing the shape of this input, the feature maps are flattened. The final CNN output is represented by the FC layer's output. As shown in Figure 3

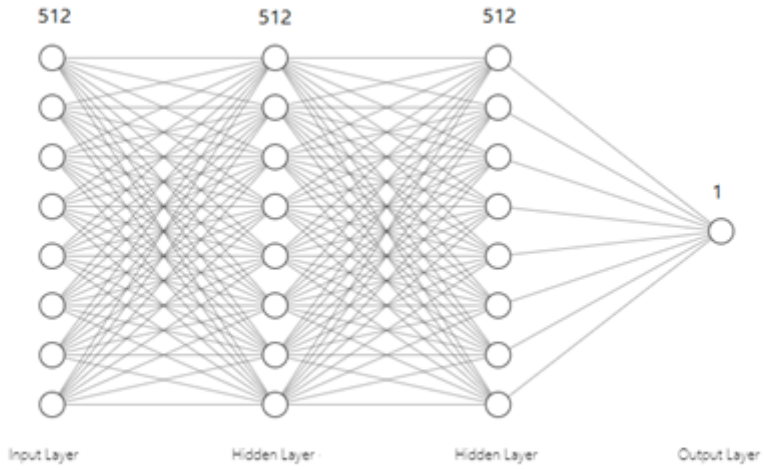

Figure 3 Fully Connected Layer

Evaluation Metrics

Determining the effectiveness of trained classifiers or learning algorithms on various data sets is the aim of assessment in deep learning. Most existing metrics concentrate on the accuracy of a classifier's class identification. Metrics for evaluating classification performance are essential for guiding the construction of classifiers. There are significant disadvantages to even the most widely used techniques, including figuring the accuracy or error rate on a test set. Consequently, optimizing criteria is correlated with modifications to classification algorithms to some extent. A lot of work has gone into creating more intricate algorithms to deal with the categorization issue. At least as important as the algorithm is the evaluation metrics phase, which comes first in the learning process.

Classifier performance may be measured using two different approaches: graphical and numerical. A classifier's performance is expressed as a single number in numerical assessments, but performance is shown on a plot with just two or three dimensions in graphical approaches, which makes it easier for humans to verify. While cost curves are examples of graphical methodologies, accuracy, precision, recall, and F1-Score are examples of numerical performance evaluations.

The percentage of correctly classified records to all records is referred to as "**accuracy**" [84].

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision quantifies the proportion of attack class predictions that fall into the anomaly class [84].

$$Precision = \frac{TP}{TP + FP}$$

Recall is defined as the percentage of anomalous records that are successfully identified divided by the total number of anomalous records [84].

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

The harmonic mean of accuracy and recall, or **F1 score**, is what's utilized to determine derived efficacy. A traffic categorization model has to have high recall, precision, and accuracy in order to be deemed excellent [85].

$$F1 - Score = 2X \frac{Recall\ X\ Precision}{Recall + Precision}$$

Results

Unlike previous research, we proposed a 2D-CNN that makes the most use of the dataset by employing a data augmentation approach. Prior studies trained the GIST + SVM classifier by selecting many characteristics linked to malware files. Secondly, training the 2D-CNN to enable autonomous feature extraction from pictures was proposed. Our solution outperforms these two methods in terms of performance (less than that of earlier work), using data augmentation techniques with 10 epochs. We were able to effectively use the dataset and prevent overfitting in this way as shown in table 1.

Table 1

Method	Accuracy of method
Model in [3]	96.46%
GIST + SVM [6]	92.27%
2D-CNN [6]	93.56%
Our Model	**99.12%**