COMPUTERS AND INFORMATICS
2757-8259 2022, Volume 2 Issue 2
Research Article

Malware classification with using deep learning

Makbule Damla Yılmaz
Gazi University, Technology Faculty, Computer Engineering Department, Ankara, Türkiye
damlaayiilmaz@gmail.com

Submitted: *21.07.2022*
Accepted: *31.12.2022*
Published: *31.12.2022*

Abstract:

Today the use of electronic devices, which phones, computers, tablets, etc. has become more and more widespread. As a result of this situation, there is a great increase in the time spent on the internet. Although the widespread use of wireless communication brings about easier access to information, it can sometimes turn the security of data into a threat by malicious people. Malware that threatens information security can cause damage to electronic devices by damaging them, stealing personal information, loss of data of large companies, and causing financial and moral damages to users. For this reason, it has become more important to ensure the security of information while people share many data on the internet uncontrollably. Artificial intelligence methods, which have been developing rapidly in recent years, will undoubtedly become an indispensable part of information security in the near future. In this study, it is aimed to detect and classify malware families by using the Convolutional Neural Networks method, which is in the deep learning subfield of artificial intelligence.

Keywords: *Artificial intelligence, Deep learning, Malicious software, Malware detection, Convolutional neural network*

Cite this paper as: Yılmaz, M. D. Malware classification with using deep learning, *Computers and Informatics*, 2022, *2*(2), 21-40.

1. INTRODUCTION

In recent years, the widespread use of the internet, information-based systems and increase in digital addiction day by day have led to the development of malicious software. Malware is software developed by malicious people, aimed at directly targeting devices, hijacking systems with various attack types, accessing other devices on the network used, or maintaining its own existence by locating some files. Malware can bring financial gains to the attacker while harming the internet user. Malware can steal personal information and use it as a threat, cause serious damage by leaking, deleting or encrypting data of national or international importance, from large-scale companies to government-affiliated institutions, causing important damage and as a result unfair financial gain.

The malware initially appeared as a form of cybervandalism. That is, it was used to change the computer background and access personal information. Since then these methods adopted by cybercriminals, have begun to be used to track information from storing valuable, commercial or personal data for ransom to stealing identities to gain access to bank accounts. Today, there are many types of viruses, especially "Ransomware", "Trojan Horse", "Worms", "Spyware", "Ad-Based Software". The common feature of all these software is that it provides unauthorized access to the systems.

The spread of malware has increased tremendously in the last 10 years, especially after the COVID-2019 pandemic. In the reports prepared every year by the important companies of cyber security, it can be observed how fast the malware spreads day by day and its devastating effect. According to the research of the AV-Test Institute an organization that evaluates and rates antivirus and security packages, there are more than 1 billion malware programs today, 560,000 pieces of malware are detected every day and 17 million new malwares are registered every month [1]. Another cybersecurity technology company CrowdStirke, announced in its annual threat report that after the ransomware attacks in 2021 data leaks increased by 82% compared to 2020 [2].

Based on these data the importance of detecting and preventing malware has increased. There are several methods used for malware detection. These methods can be categorized in 3 branches as "static analysis", "dynamic analysis" and "memory-based analysis" [3]. Static Analysis examines the malware file without running the program [4]. It is the safest way to analyze malware without infecting the system. However, since it does not run static analysis code, it may be insufficient in detecting complex malware that can infect at runtime. Dynamic Analysis runs the program and monitors its behavior in an automatically protected test environment called a "sandbox". While this test environment is highly isolated, it is also secure against malware entry or penetration into the system. However, there are some methods developed by attackers against dynamic analysis. One of these is to develop malware that sleeps itself until it passes the sandbox area. Memory Based Analysis has been a remarkable method in recent years. Accordingly, it enables to detect the presence and behavior of malware by methods such as recording memory dumps and machine learning, which provide more information in terms of the structure and functioning of malware [5]. This approach can be categorized in two ways: "signature-based analysis" and "anomaly-based analysis". Signature-Based Detection ensures that a known threat is blacklisted using a unique identifier and the threat is recognized in the future [6]. It works using pre-built lists of known threats and danger indicators. If the created signature pattern is encountered again, the program can be marked as infected by comparing the danger indicators with the database or the attack signature. Signature-based detection is the most suitable method for detecting known threats but this method may be insufficient for malware detection as malware can become increasingly common, mutate as it infects and can change rapidly. Anomaly Based Detection examines system activity and classifies it as normal or abnormal. Instead of looking for known threats, an anomaly-based detection method uses machine learning to recognize how the system normally behaves. All activities in the network are then compared with the base system. In this way, a warning is provided against unknown suspicious behavior.

The latest in malware detection shows that traditional methods can only adapt to known threats. However, today it has become a necessity to use artificial intelligence in the field of information security in order to use smarter and more automated systems against the rapidly increasing malware and to be prepared for all threats that are not only known also suspicious.

Artificial intelligence is a science based on giving machines the ability to imitate humanoid behavior. It has applications in a wide range of fields from health to trade from defense industry to finance sector. Artificial intelligence is the product obtained from the purposeful use of the machine in various sectors by training.

The process of training the machine is called "machine learning". Machine learning has many uses. Image processing-computer vision is a common example of machine learning. Depending on the density of pixels in black and white or color images an object can be described as a digital image. This way, machines can make medical diagnoses from x-ray results. Natural language processing is the human-machine interaction of the language that people use among themselves or the processing of language for the purpose of interaction between people who speak different languages. Machines can convert audio to text and text to audio. Thus translation applications, voice assistants, operator conversations can be done with machine learning. Recommendation systems are systems that aim to suggest products

that may be of interest to users based on their previous choices. These systems are used to improve the user's online service experience from shopping to advertising. Apart from these, there are applications of machine learning in many fields such as robotics and autonomous systems, credit and risk assessment systems, emotion recognition and games designed with augmented reality. Cyber security and information security have been added to these areas in recent years.

Cyber security means the use of various measures, methods and tools to protect systems from threats and vulnerabilities and to provide users with the right services efficiently [7]. There are many studies using machine learning methods to provide information security.

In recent years, it has been pointed out that the use of artificial intelligence can offer an effective solution in providing cyber security with the increase in the number of data collected about malware and their accessibility, the performance increase in the processing power of machines. However, it has been shown that artificial intelligence carries some risks as well as good aspects in cyber security. This risk has been defined as artificial intelligence facilitates the attack process as it enables better targeted, faster and more effective attacks [8][9]. On the other hand, artificial intelligence has led to increased use of machine learning to perform malware analysis along with network anomaly detection [10]. Today, machine learning can provide malware detection, intrusion detection, phishing detection, spam and fraud detection [11].

Artificial intelligence-based solutions also show success in increasing the performance of cyber security studies. For example, according to Stanford University's AI Index 2019 Report, the time required to train a large-scale image classification system on cloud infrastructure has decreased from about three hours in October 2017 to about 88 seconds in July 2019 [12].

Machine learning can basically be categorized as “supervised learning”, “unsupervised learning” and “reinforcement learning”. Machine learning is used to solve problems such as classification, regression, clustering. Classical machine learning has algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, Naive Bayes, k-Nearest Neighbor, k-Means. Deep learning is a machine learning method that includes artificial neural networks. There are many types of algorithms in deep learning, mainly Convolutional Neural Networks, Recurrent Neural Networks, Long-Short-Term Memory. Studies have been carried out with different algorithms to provide information security with artificial intelligence.

Some of the studies and achievements in the literature using various algorithms of machine learning and deep learning are as follows:

Martin et al. used the Logistic Regression and Random Forest algorithms of machine learning on Android-based malware detection in a comparative way. According to the results of the study, which was carried out with 260 thousand malware signatures and 80 thousand Android applications, approximately 89% success was achieved in Logistic Regression, while the Random Forest algorithm had an accuracy of 92% [13].

Firdasu et al. compared the performances of machine learning using k-Nearest Neighbor, Naive Bayes, Decision Tree, Support Vector Machines and Multilayer Perceptron Neural Network algorithms with dynamic analysis approach. The best results were obtained in the J48 Decision Tree algorithm with 96.8% accuracy [14].

Li et al., in their study using Support Vector Machine, Decision Tree and Random Forest algorithms, obtained over 90% sensitivity and accuracy with the Support Vector Machine [15].

Narudin et al. worked with k-Nearest Neighbor, Naive Bayes and Random Forest algorithms using anomaly-based approach for mobile malware. Naive Bayes and Random Forest algorithms gave the best results with 84.57% true accuracy against k-Nearest Neighbor Android malware, while Multilayer Neural Networks exceeded 99.97% true positive rate [16].

Jang et al., using the Convolutional Neural Network algorithm with the Microsoft Malware Classification Challenge BIG2015 dataset, achieved 99.6% accuracy [17].

Kang et al. compared Opcode, API function classification and one-hot encoding methods using the Microsoft Malware Classification Challenge BIG2015 dataset. They achieved an accuracy of 97.59% using the Long Short-Term Memory deep learning model [18].

Yan et al. detected whether the Windows executable files were malicious in their malware detection method, which they called MalNet. They achieved 99.36% accuracy using Convolutional Neural Network and Long Short-Term memory deep learning approaches [19].

As can be seen from various studies using machine learning and deep learning algorithms, although machine learning works fast, deep learning has been more successful in giving high accuracy results [20]. While classical machine learning methods are successful on small input datasets, their performance starts to decrease as the number of data increases. Deep learning is ideal for large, image datasets and provides high accuracy [21].

In this study, Dumpware10 dataset was studied with the "Convolutional Neural Network" approach, which is one of the deep learning algorithms, which is a very successful method for the detection of malware and the study was completed with an accuracy of 94%. The previous studies on the Dumpware10 dataset and the achievements of these studies are as follows:

Tekerek et al. worked with Convolutional Neural Network model on Microsoft Malware Classification Challenge BIG2015 and DumpWare10 datasets and found 99.86% for BIG2015; They achieved an accuracy of 99.60% for Dumpware10 [22].

Bozkır et al., conducted studies on the DumpWare10 dataset with Random Forest, Sequential Minimum Optimization, Support Vector Machine and XGBoost approaches [23].

Wong et al. worked with Malimg, MaleVis, Virus-MNIST and Dumpware10 datasets and reached 99.14%, 95.01%, 86.36% and 96.62% accuracy for Dumpware10, respectively [24].

2. METHODOLOGY

2.1. Research Design

Convolutional Neural Networks are used as a solution to image processing and classification problems in this study to detect malware.

Convolutional Neural Network is a known deep learning architecture inspired by the natural visual perception mechanism of living things [25]. The definition of neocognitron was made by Kunihiko Fukushima in 1979, after the idea of giving machines the ability to think and learn just as human intelligence does. Neocognitron is the name given to hierarchical and Multilayer Artificial Neural Network. It was used in tasks such as character recognition, pattern recognition with Japanese

handwriting, and inspired the Convolutional Neural Network. However, the main source of Convolutional Neural Networks is the design that allows computers to recognize handwritten numbers called LeNet, which was developed by LeCun and his team between 1989-1998.

Later, since 2012, classical and modern architectures such as AlexNet, ZFNet, VGGNet, GoogLeNet, ResNet, DenseNet, Inception have emerged. As a result, Convolutional Neural Networks are a sub-branch of deep learning and are Artificial Neural Networks that are generally used in the analysis of visual data [26].

The basic unit of a single-layer Artificial Neural Network is called a "sensor". An Artificial Neural Network (or simply neural network) consists of an input layer of neurons or nodes, one or more layers of hidden neurons and a final layer of output neurons [27].

If the neural network does not contain hidden layers as in Figure 1 and consists of only input and output layers, it is called “single-layer neural network”, if there are hidden layers as shown in Figure 2, it is called “multilayer neural network”.

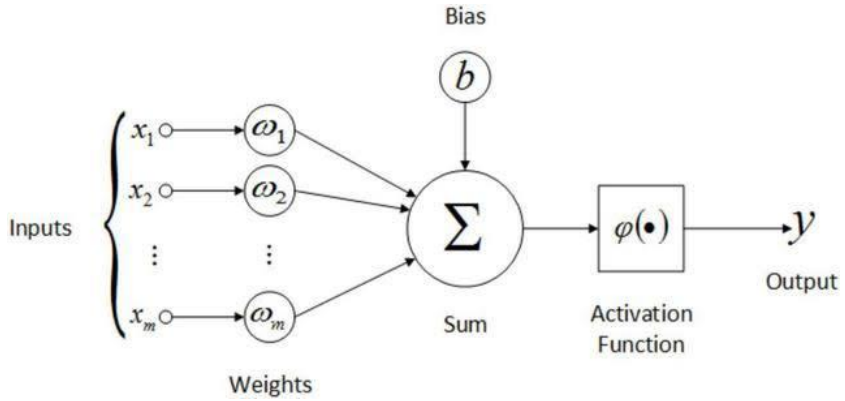


Figure 1. Single layer neural network.

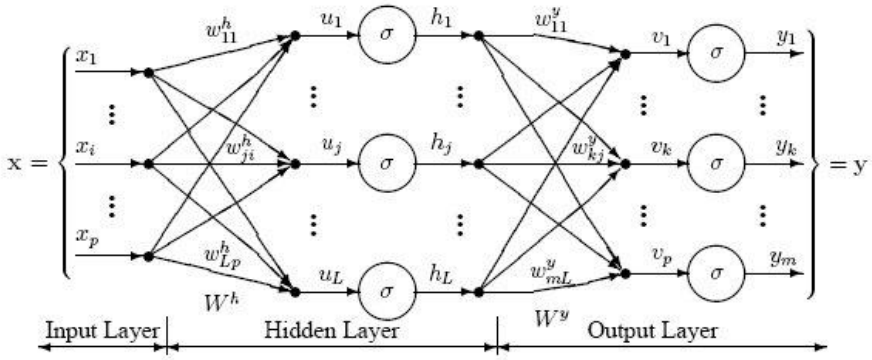


Figure 2. Multilayer artificial neural network.

The formula for the output expressed as y of a neural network that takes n inputs and has a single output layer, where the value of the inputs is expressed as x, the weights of the neurons are expressed as w, is as follows.

$$y = f\left(\sum_{i=1}^{n}(\omega_i x_i) + b\right) \tag{1}$$

Forward propagation is the computation and storage of intermediate variables sequentially from the input layer to the output layer in a neural network. Back propagation, on the other hand, is to rearrange the weights in the neural network based on the error rate obtained after iteration completed during neural network training. During back propagation training, after each iteration, the loss function is looked at and the gradient is calculated for the input-output samples according to the weights of the network. This process minimizes the loss in training multilayer networks and keeps the weights up to date. Forward and backward propagation operations are repeated until the loss function is minimized. This process is called "learning".

This learning process is based on the chain rule. The chain rule is used to calculate derivatives of functions consisting of a combination of functions whose derivatives are known. The chain rule is applied recursively to achieve backpropagation [28]. The output, which is the sum of the weight and input values and the addition of the bias value, is subjected to an activation function. This is the value that creates the output of the output layer. The activation function defines how the weighted sum of the input is converted to an output from a node or nodes in the network layer (hidden layers and output layer). Popular types of activation functions are: Sigmoid, Hyperbolic Tanh, ReLU, Leaky ReLU, Softmax, Linear, etc.

Activation function selection is often ReLU, Sigmoid and Hiberbolic Tanh in hidden layers, while Linear, Logistic and Softmax functions are preferred in the output layer [29]. In this study, ReLU activation function is used for the hidden layers and Softmax activation function is used for the output layer. In Figure 3, the value ranges of popular activation function types can be observed on the same graph.

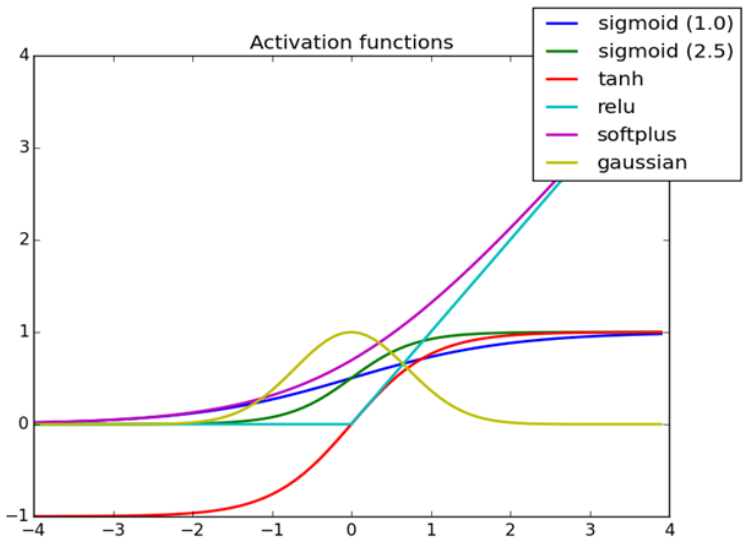


Figure 3. Comparative representation of activation functions.

A basic Convolutional Neural Network architecture consists of the convolution layer, the pooling layer and the full link layer. A stack can be created by repeating these layers more than once. Convolution and pooling layers can be 2 or 3 dimensional [30].

2.2. Convolution Layer

The convolution operation is a procedure in which two sources are intertwined, transforming one function into another [31]. This happens through filters smaller than the input size. Convolution operations are performed by applying filters to the inputs as seen in Figure 4. The convolution operations are completed by the activation function. The output, which consists of convolution with the same filter applied repeatedly to an input image, is called a "feature map". The feature map is the output of each

layer and shows the location of the features found by the Convolutional Neural Network model relative to a given input image.

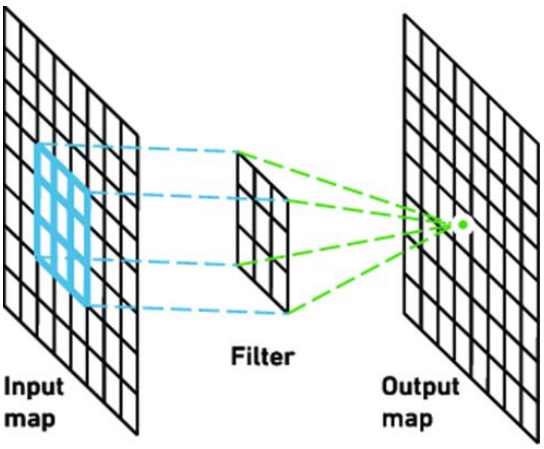


Figure 4. Convolution operation.

2.3. Pooling Layer

The task of this layer, which is added between the convolution layers, is to reduce the displacement size of the representation, learnable parameters in the network, and the number of computations [32]. It is performed in two ways: Average Pooling and Maximum Pooling. The main purpose of both is to reduce the tolerance for small variations of the input for all cases by reducing the size and to make the model generalizable. In Figure 5, the maximum pooling process is given as an example. Here, the input is divided into 4 equal parts and after the pooling process, all of these parts are expressed with the pixel value that has the highest value in itself.

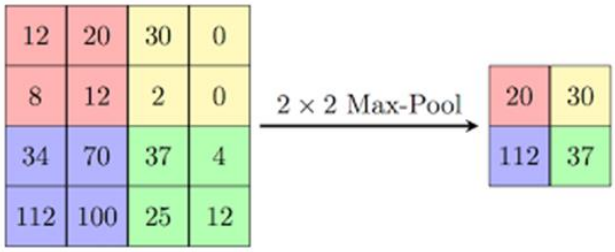


Figure 5. Maximum pooling process.

2.4. Fully Connected Layer

The fully connected layer takes the outputs from the previous layer as input, flattens the inputs into a vector for the next layer. After the convolution operations, the output should be obtained by separating the data into classes. It assists the learning process by linearizing non-linear high-level features from fully connected layer convolution operations. A fully connected layer is often added to make the model end-to-end trainable. Figure 6 shows the internal structure of the fully connected layer.

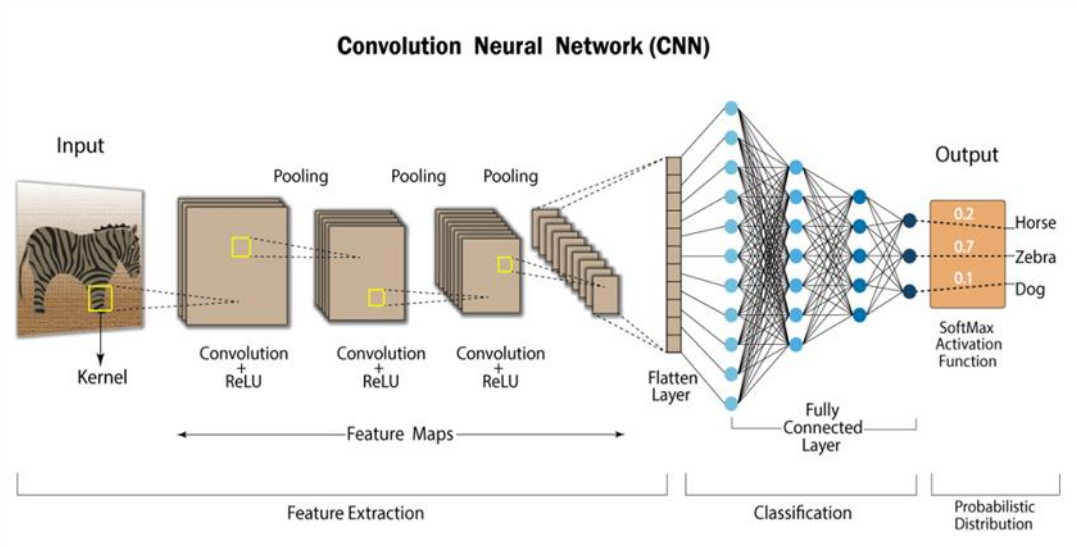


Figure 6. Fully connected layer.

Apart from these layers, there are other methods used in Convolutional Neural Networks that make the model better. For example, adding pixels, finding edges, step shifting, adding dropout layers, batch normalization, etc. The performance of the model can be updated with hyperparameter changes. In Figure 7, the layer structures of LeNet, which is a Convolutional Neural Network model are shown in detail.

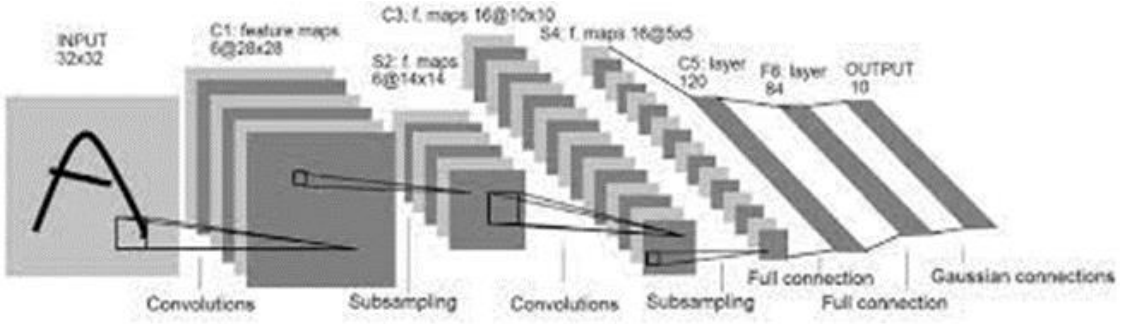


Figure 7. Appearance of LeNet architecture.

2.5. Research Universe/Dataset

The Dumpware10 dataset, which was developed in the Department of Computer Engineering at Hacettepe University in 2020, was used in the project for malware detection [33].

Dumpware10 is a dataset developed for memory dump-based malware detection and consists entirely of images.

Figure 8. Random images of classes found in the Dumpware10 dataset.

Datasets used to detect malware may be prepared at the byte level or created from ready-to-use images. While the datasets prepared at the byte level are used for deep learning algorithms such as classical machine learning and Long Short-Term Memory, they must first be transformed into images through a preprocessing process to be used in Convolutional Neural Network algorithms. These images can consist of 3-channel color images in the form of gray or RGB (Red-Green-Blue). Dumpware10 dataset consists of 3 channels of 4 different size images in RGB format as seen in Figure 8. Dimensions are 224, 300, 4096 and square root. In the study, 300x300 images found under the 4096 folder were used [34].

The Dumpware10 dataset consists of 11 malware classes, 10 malicious and 1 benign [35]. It consists of 4294 RGB images in total, 3686 harmful and 608 benign. Malware families and class distributions in the dataset are visualized with a bar graph in Figure 9. The graphs were created using Python's Matplotlib library, which is used for visualization.

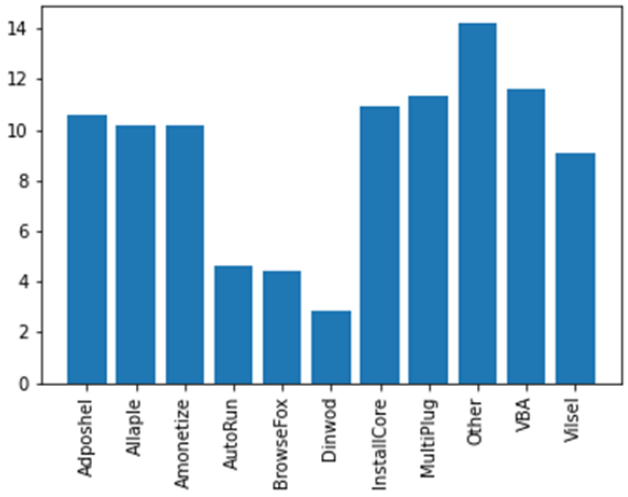


Figure 9. Distribution of malware classes in the dataset.

In addition, RGB data in the Dumpware10 dataset are separately foldered for training and testing. The distribution ratios of the data in the folders are shown in Figure 10.

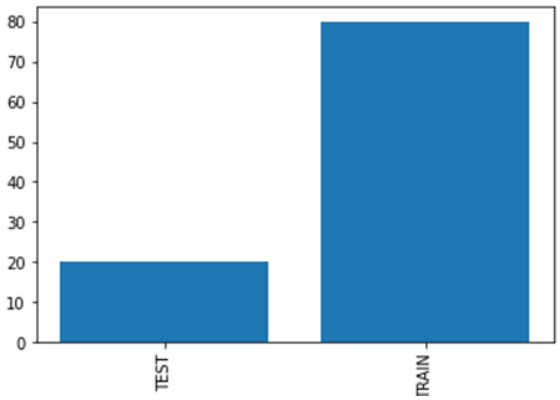


Figure 10. Bar graph of training and test folders.

2.6. Architecture

In this study, Keras library, which is TensorFlow's high-function API, was used. TensorFlow is an open-source library developed by Google and written in Python for use in deep learning studies.

Before building a basic Convolutional Neural Network architecture, the dataset must be included in the program. Dumpware10 dataset, consisting of 3-channel RGB format images, was included in the program codes as it was ready to use.

Keras' ImageDataGenerator() method is used to work with data in image format. In this method, the size of all images in the dataset can be rearranged and the data can be processed in the same size or the performance can be accelerated by saving time by reducing the size. Then, each image in the dataset is processed and labeled one by one, as seen in Figure 11. RGB data is clipped to input data in the range [0..1] for float type and [0..255] for integer type with the imshow() method.

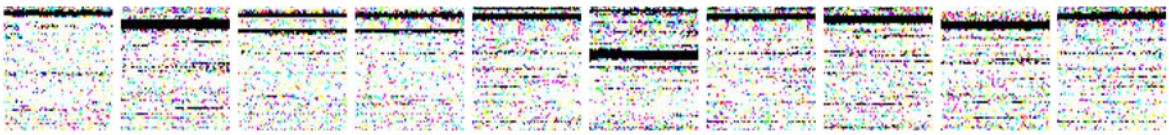


Figure 11. RGB data display.

After the RGB data is labeled one by one and converted from the image to an array, the model building step is started. Since Convolutional Neural Network models consist of layers that must be added sequentially, they are usually set up by calling Keras' Sequential() function. Convolutional layers, pooling layers, fully connected layer and output node are added inside the Sequential() method. In this study, 2D convolution and pooling layers are used.

First, the convolution layer, expressed as Conv2D(), is added. Filters used to select features are included in this layer. The number of filters is usually used incrementally at each convolution layer. Filters can be used as many as the number of features desired to be selected. The size of the filter is specified with the "kernel_size" argument.

When filtering is applied to the input matrix, the dimensions change according to the matrix multiplication and since the filter size will always be smaller than the image size, the output matrix will experience a size reduction. In order to prevent size reduction, pixels are added to the edges and corners of the output matrix by applying pixel addition. Thus, the input and output matrix sizes are equalized.

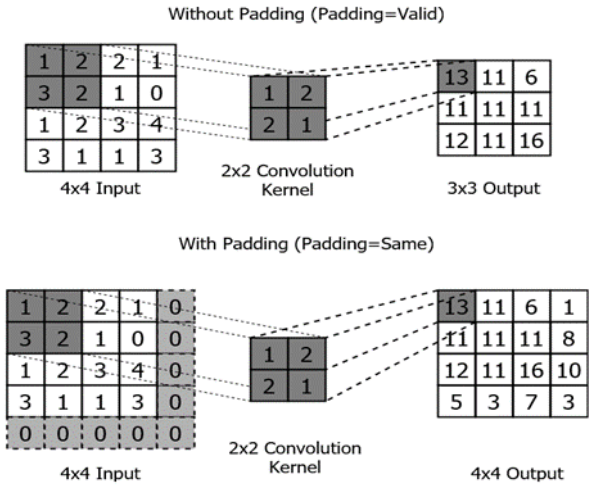


Figure 12. Effect of pixel addition on output matrix.

As seen in Figure 12, adding pixels continues with the "same" argument and performs this operation, ensuring that the input matrix and output matrix are the same size. The pixel addition method can be done by copying the values found in the last frame of the output matrix. Although the processing load increases, more precise and close outputs are obtained.

Basically, in a Simple Artificial Neural Network, x is defined as inputs, w as weights and the activation process is applied to f(x), which is the value transferred to the output of the network. Then this will be the final output or the input of another layer [36]. If the activation function is not applied to the neural networks, a linear function is obtained as an output. But the main problem of neural networks is complex nonlinear real-life data (sound, image, text, etc.). For this reason, the ReLU activation function can be used to preserve this property of non-linear data.

ReLU(Rectified Linear Unit) is an activation function that enables learning from complex data that appears linear but is not linear to use back propagation and stochastic gradient descent to train deep neural networks. As seen in Figure 13, if the ReLU input is positive (greater than 0) it will output as it is, if it is negative (less than 0) it will output 0.

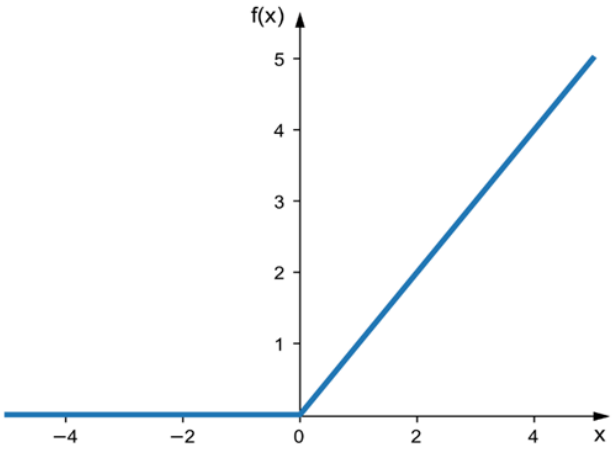


Figure 13. ReLU activation function.

In order for image data to be included in convolution operations, its dimensions must be given in the "input_shape" argument. For example, if the images are in RGB format with a size of 300x300, the input size (300x300x3) should be given. The number 3 at the end indicates that the images are in RGB format consisting of 3 channels of red-green-blue colors. It is specified only in the first convolution layer.

After the convolution layer is completed, it is passed to the pooling layer. In this layer, the size of the pooling layer is selected with the "pool_size" argument. Step shifting can be done with the "stride" parameter. For example, if 2 is given in the stride, the filters are shifted at intervals of 2 pixels.

After the convolution and pooling layers are repeated several times, the dataset is converted into a 1-dimensional array with the fully connected layer. This output forms the input of the neural network. There should be as many neurons as how many classes are used in the output layer. For example, since there are 11 classes in the dataset, the number of neurons in the output layer is 11. "Softmax" is a commonly used activation function in the output layer. Softmax function is a function that converts a number vector to a probability vector. After creating the architecture of the model, training processes are started. This process is completed in three steps: compiling the Convolutional Neural Network model, training the model and evaluating the results. First, the compile() method is called to compile the model. Metric selection is made in order to evaluate the optimization algorithm, learning rate, loss function and the performance of the model to be used in this method.

In this study, "Adam" was chosen as the optimization algorithm. The optimization algorithm provides the best value in non-linear situations. Other algorithms such as Gradient Descent, Stochastic Gradient Descent, Adagrad, Adadelta can also be used.

A learning rate is determined together with the optimization algorithm. The learning rate is a parameter used to adjust the model weights according to the change in the subduction function. Choosing a small learning rate can cause the model to undergo a very slow and difficult training process, while large learning rates can result in fast but unstable training.

The loss function mostly measures the relationship between the predicted value and the actual value. The loss function is highly effective in neural network performance [37]. Loss function is also expressed as cost and error function. The purpose of a neural network in the training result is to increase the optimization and minimize the cost function. There are several different types of loss functions. Some of them are as follows: Error Mean Squares, Absolute Mean Error, Binary-cross Entropy, Categorical-cross Entropy, Logarithmic Loss. Since this study is a multi-class classification problem, "categorical-cross entropy" is used as a loss function. The "accuracy" metric was used to evaluate the performance of the neural network. In classification problems, other than accuracy, metrics such as precision, sensitivity and f-1 score can also be used.

After the model is compiled, training is done with the fit() function. Before the training process, a part of the dataset reserved for the model's own training can be optionally separated as the validation dataset. The algorithm is determined for the accuracy of the model in the training dataset and the model is tried to be improved by choosing the most appropriate weights in the validation dataset. The separation is done by calling the train_test_split() method from the model_selection module of Sci-kit Learn, Python's frequently used library for machine learning. The division of the data set is done randomly by giving a certain ratio. In this study, 0.3 (30%) data were reserved for validation in the training dataset. Training and validation data are run with 200 iterations. The number of iterations can be expressed as a period in which the entire training set is used to update the weights in training a model [38].

In deep learning, the performance and loss function of the model reach their best value at a certain point. If more training is continued after the number of iterations for which the model is at its best, the performance will decrease and the model may start to learn more, or if less training is done, the model will not learn enough and fail again. Therefore, choosing how long to train the model can be difficult. An early stop is used as a solution to this problem. Early stopping stops training when the loss function reaches a constancy or starts an increasing [39]. Early stopping was also used in this study. For early stopping, Keras has an object called "callback". With callback, some actions can be performed at certain

points in the training. For example, if there is no improvement in the validation loss function during 20 iterations in this study, the training is stopped.

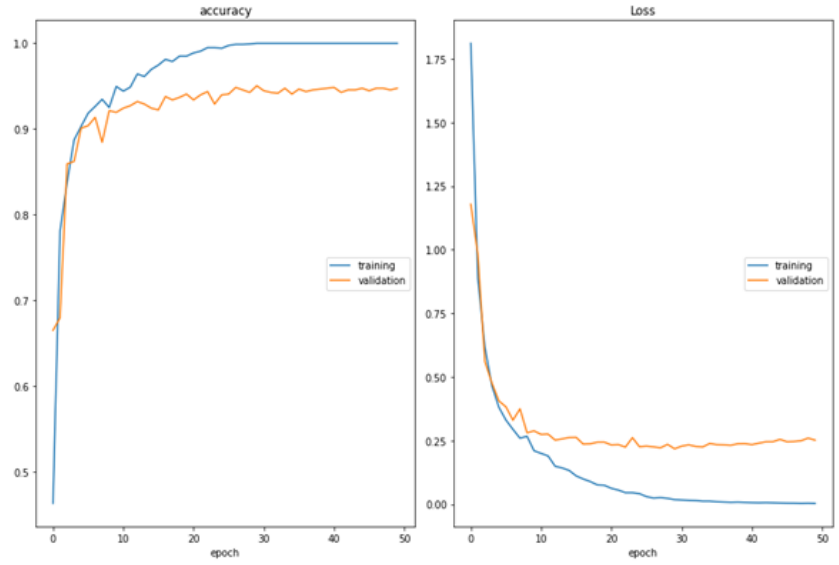
3. FINDINGS

The architecture of the Convolutional Neural Network model developed with Keras on the TensorFlow backend is given in Figure 14.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 32)        896

 max_pooling2d (MaxPooling2D  (None, 32, 32, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 32, 32, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 128)               2097280

 dense_1 (Dense)             (None, 11)                1419

=================================================================
Total params: 2,118,091
Trainable params: 2,118,091
Non-trainable params: 0
_________________________________________________________________
```

Figure 14. Convolutional neural network model.

Two two-dimensional convolution layers, two two-dimensional maximum pooling layers and a fully connected layer were added to the model. 128-node hidden layer and 11-node output layer were added to the output of the convolutional architecture. With the validation data separated by 0.3 ratio from the train dataset, model training with 200 iterations was started. Since the process takes too long and a very different change in model performance is observed when going up to 200 iterations, the early stop feature has been added to the model. In this way, model was automatically stopped at the 50th iteration, as no further improvement in the validation loss value occurred. With the "Livelossplot" feature, the accuracy, validation accuracy, loss and validation loss values of the model were plotted live during the training in each iteration. Graphics are given in Figure 15.




```
accuracy
        training                (min:    0.463, max:    1.000, cur:    1.000)
        validation              (min:    0.665, max:    0.950, cur:    0.948)
Loss
        training                (min:    0.003, max:    1.811, cur:    0.003)
        validation              (min:    0.217, max:    1.179, cur:    0.251)
76/76 [==============================] - 18s 239ms/step - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.2514 - val_accuracy: 0.9476
Epoch 50: early stopping
<keras.callbacks.History at 0x7f2161cb54d0>
```

Figure 15. Accuracy and Loss graphs.

At the end of the training, the accuracy of the training dataset reached 1,000, while the accuracy of the validation dataset reached 0.948. During training, the validation dataset showed up to 95% accuracy as the best rate.

It was rerun with a few different adjustments to improve the model by looking at the change of some hyperparameters and to see the differences.

Unlike the first model, it was observed how much the model was affected by this node by deleting the 128-node hidden layer. Architecture is given in Figure 16 and graphic visualizations are given in Figure 17.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 32)        896

 max_pooling2d (MaxPooling2D  (None, 32, 32, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 32, 32, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 11)                180235

=================================================================
Total params: 199,627
Trainable params: 199,627
Non-trainable params: 0
_________________________________________________________________
```

Figure 16. Convolutional neural network model.

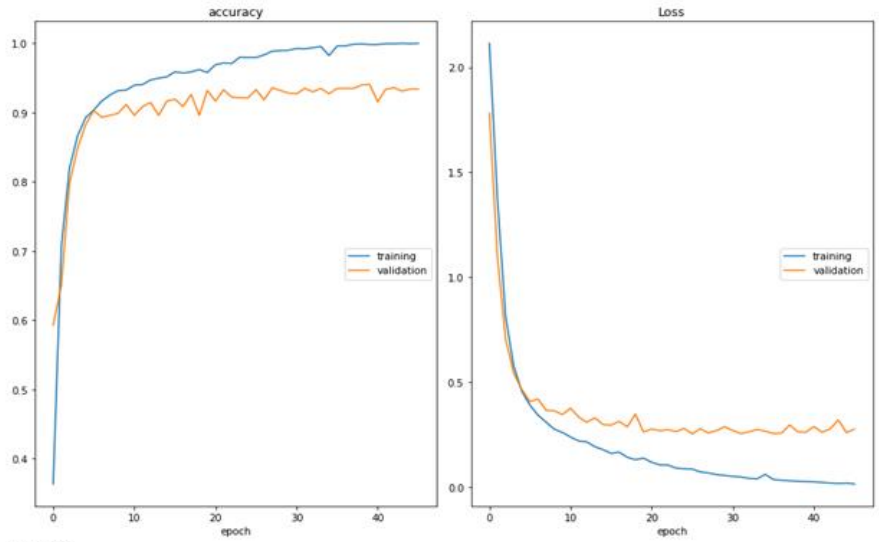


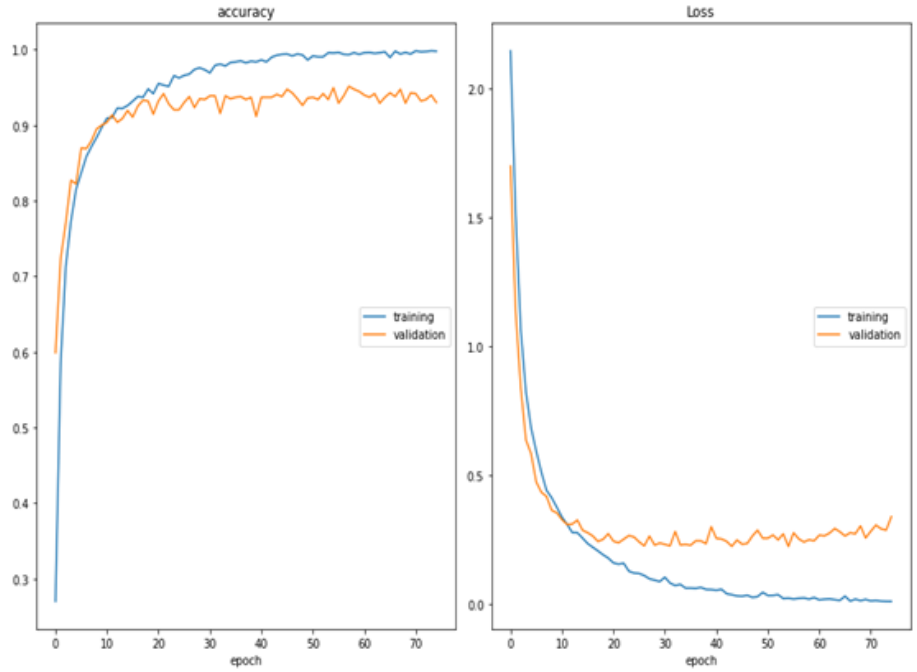
Figure 17. Accuracy and Loss graphs.

The training was stopped at the 46th iteration. This model was slightly less successful than the first model.

In another model, a dropout layer was added in which some nodes were deleted in order to prevent over-learning of the model. A forgetting rate of 0.25 was determined in the first drop layer and 0.3 in the second layer. In addition to 128-node hidden layer, 64-node hidden layer has been added and with 200 iterations and early stopping feature, all the remaining parameters have the same value. Training 75th iteration is over. The summary of the model is given in Figure 18 and the loss and accuracy graphs are given in Figure 19.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 32)        896

 max_pooling2d (MaxPooling2D  (None, 32, 32, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 32, 32, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 dropout (Dropout)           (None, 16, 16, 64)        0

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 128)               2097280

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 11)                715

=================================================================
Total params: 2,125,643
Trainable params: 2,125,643
Non-trainable params: 0
_________________________________________________________________
```

Figure 18. Convolutional neural network model.




```
accuracy
        training                (min:    0.270, max:    0.998, cur:    0.998)
        validation              (min:    0.599, max:    0.951, cur:    0.930)
Loss
        training                (min:    0.012, max:    2.148, cur:    0.012)
        validation              (min:    0.224, max:    1.701, cur:    0.340)
76/76 [==============================] - 18s 242ms/step - loss: 0.0115 - accuracy: 0.9979 - val_loss: 0.3404 - val_accuracy: 0.9301
Epoch 75: early stopping
<keras.callbacks.History at 0x7f21d21d3b90>
```

Figure 19. Accuracy and Loss graphs.

Looking at the outputs of the models, it was observed that the model without dropout layer was more successful than the model with more hidden layers, and the model with more hidden layers had higher performance than the model with less hidden layer.

3.1. Evaluation Metrics

Evaluation metrics are used to measure the performance of a predictive statistical model or machine learning model [40]. There are many different metrics that can be used to test a model. For this reason, metric selection is an important issue in order to accurately show the quality of the model.

A confusion matrix outputs another matrix and describes the full performance of the model [41]. Also known as "error matrix". It is a way to visualize the performance of the algorithm as in Table 3.1.

Table 1. Confusion matrix.

		Actual Class	
		Positive (**P**)	Negative (**N**)
Predicted Class	Positive (**P**)	True Positive (**TP**)	False Positive (**FP**)
	Negative (**N**)	False Negative (**FN**)	True Negative (**TN**)

Accuracy is the ratio of correctly predicted values to the total values.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (2)$$

Error is the ratio of incorrectly estimated values to the total values.

$$Error = \frac{FP + FN}{TP + FP + TN + FN} \quad (3)$$

Sensitivity-Precision indicates how well the positive class is predicted. They have the same equation.

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

Precision is the ratio of true positive values to all positive estimates.

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

The F-1 score combines the harmonic average of the precision and recall values into a single metric and provides a balanced value from the combination of the two.

$$F - Score = \frac{2 * precision * recall}{precision + recall} \quad (6)$$

From the metrics module of the Sci-kit Learn library, these metrics are accuracy_score, precision_score, etc. methods can be accessed and used. The results in this study model are as follows:

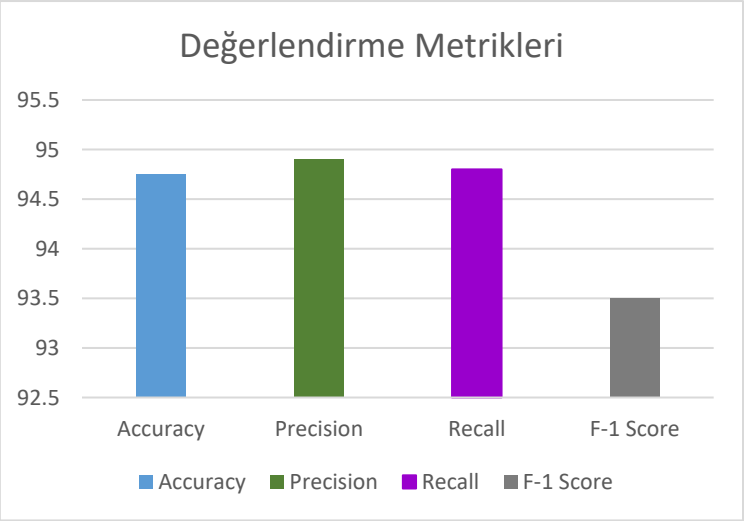


Figure 20. Visual comparison of evaluation metrics.

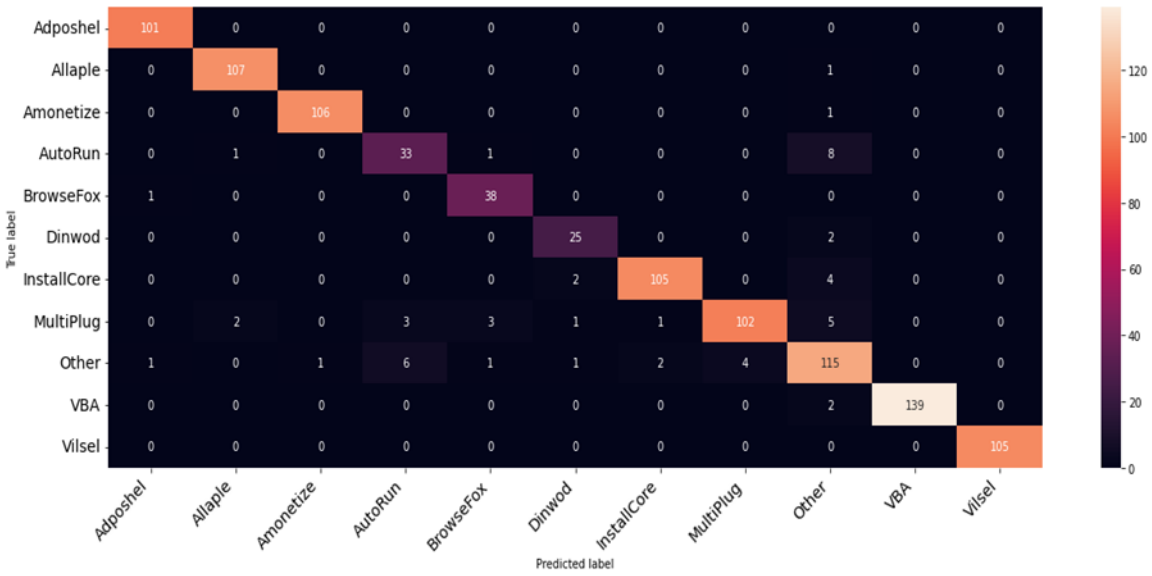


Figure 21. Confusion matrix.

There are 3 studies in the literature with the Dumpware10 dataset. With this dataset, there are two malware detection studies using Convolutional Neural Network architecture and a malware detection using various machine learning algorithms.

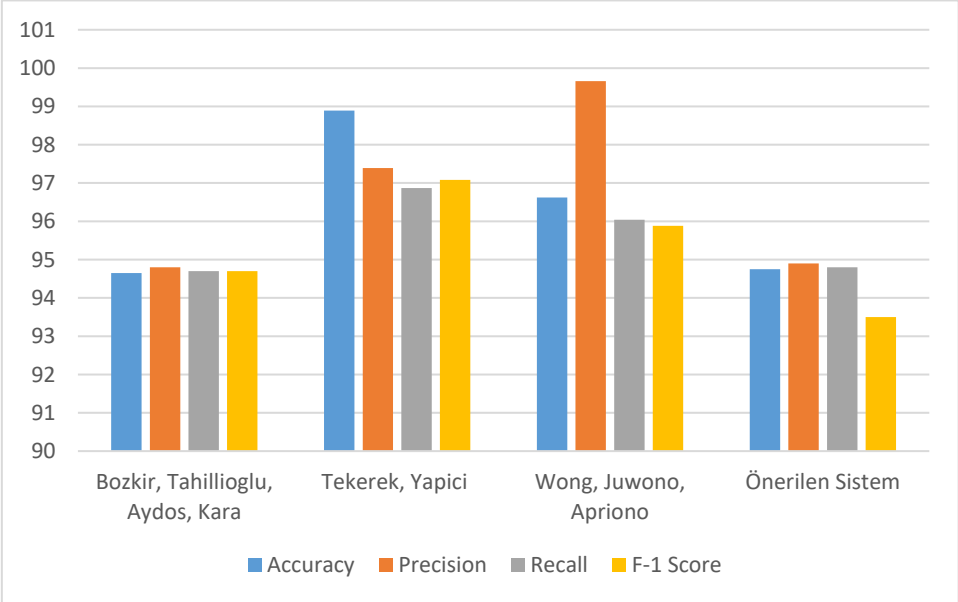


Figure 22. Visual comparison of performance with other studies in the literature.

In Figure 22, the study by Bozkir et al. was done with various machine learning algorithms and the best scores were visualized in all studies.

4. CONCLUSION

A result close to the limited literature studies made on the Dumpware10 dataset using the deep learning malware detection project and Convolutional Neural Network algorithm has been revealed. Today, while the usage areas of artificial intelligence are increasing day by day, studies are carried out for the

detection of malicious software. Although a large part of the literature was tried to be examined during this project, the fact that information security in artificial intelligence is a rarer field of study compared to other fields created difficulties for the search for guides on some sources and methods. In the study, image classification was performed for 11 different virus families by using two-dimensional convolution and pooling layers, dropout layer, fully connected layer, hidden layers and output layer. During the study, it is aimed that the model will achieve the best result in the selection of hyperparameters such as activation, loss and optimization functions. In addition, the optimum number of trainings was applied by using the necessary mechanisms in order to avoid deficiency or overfitting in the model training. As a result, a study with 94.75% accuracy, 94.9% precision, 94.8% sensitivity and 93.5% F-1 score emerged.

REFERENCES

[1] Jovanovic, B. A Not So Common Cold: Malware Statistics in 2022. 2022. <https://dataprot.net/statistics/malware-statistics/>

[2] Cook S. Malware statistics and facts for 2022. 2022. <https://www.comparitech.com/antivirus/malware-statistics-facts/>

[3] Sihwail, R., Omar, K., Zainol Ariffin, K. A., Al Afghani, S. Malware Detection Approach Based on Artifacts in Memory Image and Dynamic Analysis. 2019; 9(18), 3680. <https://doi.org/10.3390/app9183680>

[4] Tekerek, A. A novel architecture for web-based attack detection using convolutional neural network. 2021; 100, 102096. <https://doi.org/10.1016/j.cose.2020.102096>

[5] Bozkir, A. S., Tahillioglu, E., Aydos, M., Kara, I. Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. 2021; 103, 102166. <https://doi.org/10.1016/j.cose.2020.102166>

[6] Tekerek, A., Yapici, M. M. A novel malware classification and augmentation model based on convolutional neural network. 2022; 112, 102515. <https://doi.org/10.1016/j.cose.2021.102515>

[7] Zhang, Z., Ning, H., Shi, F., Farha, F., Xu, Y., Xu, J., Zhang, F, Choo, K. K. R. Artifcial intelligence in cyber security: research advances, challenges, and opportunities. 2022; 55, 1029-1053. <https://doi.org/10.1007/s10462-021-09976-0>

[8] Taddeo, M. Three Ethical Challenges of Applications of Artificial Intelligence in Cybersecurity. 2019; 29. <https://doi.org/10.1007/s11023-019-09504-8>

[9] Gibert, D., Mateu, C., Planes, J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. 2020; 153. <https://doi.org/10.1016/j.jnca.2019.102526>

[10] Alhayani, B., Mohammed, H. J., Chaloob, I. Z., Ahmed, J. S. Effectiveness of artificial intelligence techniques against cyber security risks apply of IT industry. 2021. <https://doi.org/10.1016/j.matpr.2021.02.531>

[11] Shaukat, K., Luo, S., Varadharajan, V., Hameed, I. A., Chen, S., Liu, D., Li, J. Performance Comparison and Current Challenges of Using Machine Learning Techniques in Cybersecurity. 2020; 13(10)-2509. <https://doi.org/10.3390/en13102509>

[12] Zhang, Z., Ning, H., Shi, F., Farha, F., Xu, Y., Xu, J., Zhang, F., Choo, K. K. R. Artifcial intelligence in cyber security: research advances, challenges, and opportunities. 2022; 55, 1029-1053. <https://doi.org/10.1007/s10462-021-09976-0>

[13] Martín, I., Hernández, J. A., de los Santos, S. Machine-Learning based analysis and classification of Android malware signatures. 2019; 97. <https://doi.org/10.1016/j.future.2019.03.006>

[14] Firdausi, I., Lim, C., Erwin, A. Analysis of machine learning techniques used in behavior-based malware detection. 2010; 2. <https://doi.org/10.1109/ACT.2010.33>

[15] Li, J., Sun, L., Yan, Q., Li, Z., Srisa-an, W., Ye, H. Significant Permission Identification for Machine Learning Based Android Malware Detection. 2018; 14. <https://doi.org/10.1109/TII.2017.2789219>

[16] Narudin, F. A., Feizollah, A., Anuar, N. B., Gani, A. Evaluation of machine learning classifiers for mobile malware detection. 2014; 20. <http://dx.doi.org/10.1007/s00500-014-1511-6>

[17] Jang, S., Li, S., Sung, Y. FastText-Based Local Feature Visualization Algorithm for Merged Image-Based Malware Classification Framework for Cyber Security and Cyber Defense. 2020; 8-460. <http://dx.doi.org/10.3390/math8030460>

[18] Kang, J., Jang, S., Li, S., Jeong, Y. S., Sung, Y. Long short-term memory-based Malware classification method for information security. 2019; 77. <https://doi.org/10.1016/j.compeleceng.2019.06.014>

[19] Yan, J., Qi, Y., Rao, Q. Detecting Malware with an Ensemble Method Based on Deep Neural Network. 2018; 2018. <http://dx.doi.org/10.1155/2018/7247095>
[20] Tekerek, A., Yapici, M. M. A novel malware classification and augmentation model based on convolutional neural network. 2022; 112, 102515. <https://doi.org/10.1016/j.cose.2021.102515>
[21] Raj, D. D. S., Pal, D. Malware patterns detection and prediction using cloud based deep learning for secured network environment. 2021. <https://doi.org/10.1016/j.matpr.2021.01.611>
[22] Tekerek, A., Yapici, M. M. A novel malware classification and augmentation model based on convolutional neural network. 2022; 112, 102515. <https://doi.org/10.1016/j.cose.2021.102515>
[23] Bozkir, A. S., Tahillioglu, E., Aydos, M., Kara, I. Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. 2021; 103, 102166. <https://doi.org/10.1016/j.cose.2020.102166>
[24] Wong, W. K,. Juwono, F. H., Apriono, C. Vision-Based Malware Detection: A Transfer Learning Approach Using Optimal ECOC-SVM Configuration. 2021; 9. <http://dx.doi.org/10.1109/ACCESS.2021.3131713>
[25] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., Chen, T. Recent advances in convolutional neural networks. 2018; 77. https://doi.org/10.1016/j.patcog.2017.10.013
[26] Kumar, B. 2021. <https://medium.com/appyhigh-technology-blog/convolutional-neural-networks-a-brief-history-of-their-evolution-ee3405568597>
[27] Wang, S. C. Artificial Neural Network, Chapter 5. 2003; 81-100. http://dx.doi.org/10.1007/978-1-4615-0377-4_5
[28] Goodfellow, I., Bengio, Y., Courville, A. Deep Learning Book. 2018.
[29] Brownlee, J. 2021. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/
[30] Yamashita, R., Nishio, M., Do, R., Togashi, K. Convolutional neural networks: an overview and application in radiology. 2018. http://dx.doi.org/10.1007/s13244-018-0639-9
[31] Convolutional Layer. 2018. <https://databricks.com/glossary/convolutional-layer>
[32] Yamashita, R., Nishio, M., Do, R., Togashi, K. Convolutional neural networks: an overview and application in radiology. 2018. <http://dx.doi.org/10.1007/s13244-018-0639-9>
[33] https://web.cs.hacettepe.edu.tr/~selman/dumpware10/
[34] https://web.cs.hacettepe.edu.tr/~selman/dumpware10/
[35] Tekerek, A., Yapici, M. M. A novel malware classification and augmentation model based on convolutional neural network. 2022; 112, 102515. <https://doi.org/10.1016/j.cose.2021.102515>
[36] Kızrak, M. A. Derin Öğrenme İçin Aktivasyon Fonksiyonlarının Karşılaştırılması. 2019. https://ayyucekizrak.medium.com/derin-%C3%B6%C4%9Frenme-i%C3%A7in-aktivasyon-fonksiyonlar%C4%B1n%C4%B1n-kar%C5%9F%C4%B1la%C5%9Ft%C4%B1r%C4%B1lmas%C4%B1-cee17fd1d9cd
[37] Tian, Y., Su, D., Lauria, S., Liue, X. Recent advances on loss functions in deep learning for computer vision. 2022; 497. <https://doi.org/10.1016/j.neucom.2022.04.127>
[38] Amidi, A., Amidi, S. Derin Öğrenme püf noktaları ve ipuçları el kitabı. <https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks>
[39] Amidi, A., Amidi, S. Derin Öğrenme püf noktaları ve ipuçları el kitabı. <https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks>
[40] Brownlee, J. Tour of Evaluation Metrics for Imbalanced Classification. 2020. <https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/#:~:text=An%20evaluation%20metric%20quantifies%20the,values%20in%20the%20holdout%20dataset.>
[41] Evaluation Metrics <https://deepai.org/machine-learning-glossary-and-terms/evaluation-metrics>