aurum

AURUM MÜHENDİSLİK SİSTEMLERİ VE MİMARLIK DERGİSİ
AURUM JOURNAL OF ENGINEERING SYSTEMS AND ARCHITECTURE

Cilt 7, Sayı 1 | Yaz 2023
Volume 7, No 1 | Summer 2023, 11-26

MALWARE DETECTION USING DEEP LEARNING ALGORITHMS

Mohammed ALTAİY, İncilay YILDIZ[2*], Bahadır UÇAN[3]

[1]Department of Information Technology, Altınbaş University, İstanbul, Türkiye
maltaiy98@gmail.com (https://orcid.org/0000-0003-2943-3857)
[2]Department of Management Information Systems, Altınbaş University, İstanbul, Türkiye
incilay.yildiz@altinbas.edu.tr (https://orcid.org/ 0000-0001-5572-5058)
[3]Department of Communication Design, Yıldız Technical University, İstanbul, Türkiye
bucan@yildiz.edu.tr (https://orcid.org/ 0000-0003-4062-0469)

Received: 20.05.2023 Research Article
Accepted: 24.05.2023
*Corresponding author

Abstract

Background/aim: The aim of this study is to benefit from deep learning algorithms in the classification of malware. It is to determine the most effective classification algorithm by comparing the performances of deep learning algorithms.
Materials and methods: In this study, three deep learning methods, namely Long-Short-Term Memory Network (LSTM), Convolutional Neural Network (CNN), and Multitasking Deep Neural Network (DNN) were used.
Results: According to the findings obtained in malware detection, the best results were obtained from LTSM, CNN and DNN methods, respectively. With the three deep learning algorithms, the average Accuracy was 96%, the Precision average was 97%, and the Recall average was 97%.
Conclusion: According to the most effective results obtained from this study, Accuracy 0.982, Precision 0.988 and Recall 0.990.
Keywords: Cyber Security, Deep Learning, Classification

DERİN ÖĞRENME ALGORİTMALARI İLE KÖTÜ AMAÇLI YAZILIM TESPİTİ

Öz

Amaç: Bu çalışmanın amacı, kötü amaçlı yazılımların sınıflandırılmasında derin öğrenme algoritmalarından yararlanmaktır. Derin öğrenme algoritmalarının performanslarını karşılaştırarak en etkin sınıflandırma algoritmasını belirlemektir.
Materyaller ve yöntem: Bu çalışmada, Uzun-Kısa Süreli Bellek Ağı (LSTM), Evrişimli Sinir Ağı (CNN) ve Çok Görevli Derin Sinir Ağı (DNN) olmak üzere 3 adet derin öğrenme yöntemi kullanılmıştır.
Bulgular: Kötü amaçlı yazılım tespitinde elde edilen bulgularra göre en iyi sonuçlar sırasıyla LTSM, CNN ve DNN yöntemlerinden alınmıştır..Üç derin öğrenme algoritması ile Doğruluk ortalama %96, Kesinlik ortalama %97, Duyarlılık ortalama %97 bulunmuştur.
Sonuç: Bu çalışmadan elde edilen en etkin sonuçlara göre Doğruluk 0,982, Kesinlik 0,988 ve Duyarlılık 0,990 oranındadır.
Anahtar Kelimeler: Siber Güvenlik, Derin Öğrenme, Sınıflandırma

1. Introduction

In the near time all of the humans has started utilizing the Internet for their daily tasks in the last recent years. This is due to the fact that virtually everything—including social interactions, online banking, transactions involving health issues, and marketing—requires the Internet. Criminals have begun to conduct crimes online rather than in the actual world due to the Internet's fast growth. Malicious software is typically used by criminals to initiate cyber-attacks against the target PCs.

Malicious software has been around for as old as computer, such as Computer viruses, worms, Trojan horses, and malware. Infection spread two years after the debut of the first communication framework. Infection was given the

name "Creeper" after a testing run indicated a framework. If you can manage to capture me, I'm the creep. It gave accurate reporting. At the time when PC technology improved and started to take over in the world's governments, uncommon illnesses appeared to disturb its utilization. Programmer groups from all around either try to overwhelm and destroy a government-run institution as well as seize some enigmatic data. For both circumstances, some malware is usually engaged in the operation (Aycock, 2006).

Mysterious describes how the most secretive online community for political activists as technology professionals is presented (Nightingale, 2015). They frequently concentrate on large-scale DDoS (distributed denial of service) ambushes, which aim to render servers or other infrastructure resources inaccessible (Danniels, 2015).

The majority of wild malware is in both of the two major file formats for Windows and Android OS, which are portable executables and apk. The use of "unknown" and "zero day" malware, which incorporates alternative signature-based detection methods, is not recognised by traditional signature-based malware approaches. The limitations of the conventional system may be solved utilising static or dynamic attributes by employing deep learning approaches, which provide an alternate option. Due to the exponential development in the quantity of malware and its increasing sophistication, virus detection takes time. The discovery of malware is often the biggest worry for consumers and the internet. This paper focuses on the following research issue: The issue with the study. In order to enhance the identification of features and more accurately identify malware using deep learning algorithms, a malware detection system should be developed by investigating the potential of static features using function engineering approaches.

Malware detection is important for several reasons (Mao et al., 2017).

- Protecting computer systems and data: Malware, such as viruses, worms, Trojans, ransomware, and spyware, can compromise the security of computer systems and data. It can lead to unauthorized access, data breaches, financial losses, and damage to critical systems. By detecting malware, organizations can take appropriate measures to prevent or mitigate these risks.
- Preventing financial losses: Malware attacks can result in significant financial losses for individuals, businesses, and even governments. Malware can be used for various malicious activities, including stealing sensitive information, conducting fraudulent transactions, or disrupting online services. Detecting malware allows for timely actions to prevent or minimize financial damages.
- Safeguarding personal privacy: Malware often targets personal information, such as credit card details, social security numbers, login credentials, and private communications. By detecting malware, individuals can protect their privacy and prevent unauthorized access to their personal information.

- Ensuring system integrity and reliability: Malware can compromise the integrity and reliability of computer systems, causing them to malfunction, crash, or become unresponsive. By detecting malware, organizations can identify and remove these threats, ensuring the stability and performance of their systems.

Cilt 7, Sayı 1 | Yaz 2023
Volume 7, No 1 | Summer 2023, 11-26

- Preserving reputation and trust: Malware attacks can damage the reputation and trust of organizations. A security breach resulting from malware can lead to customer dissatisfaction, loss of business, and a tarnished image. By detecting and mitigating malware, organizations demonstrate their commitment to security and protect their reputation.
- Compliance with regulations and standards: Many industries have regulations and standards in place that require organizations to implement security measures and protect against malware threats. By detecting malware, organizations can comply with these requirements and avoid legal and regulatory penalties.
- Overall, malware detection is essential for safeguarding computer systems, data, privacy, finances, and reputation. It helps prevent security breaches, financial losses, and disruptions, while maintaining the integrity and trustworthiness of systems and organizations.

In this paper we will do malware detection using of Deep Learning classification by making test on public dataset using the algorithms of the above tow classification methods and we will discuss the best result I n conclusion part.

2. Theoretical Background

The task of identifying malware is one that contributes to the accurate identification and subsequent treatment of malware families. Therefore, we advise adopting artificial intelligence techniques, such as deep learning (CNN, LTSM, DNN) various classifiers. Each of these techniques calls for some analytical functionality in order to extract any pertinent data from the data. The major goal of this study is to identify and evaluate several sets of traits that are appropriate for classifying malware. And also we want to make a comparison to the various feature sets in terms of their importance for the specific objective. The performance of the chosen systems is then compared and evaluated.

For reach the planned approach, researchers take the strategy shown below (Vasan et al, 2020)

- Picking a data set: Information on traffic flow is used as the methodology's first component. Researchers could do this by excluding NetFlows from the genuine data flow of a recognised entity. Another way to use a set of publicly available data in the absence of data transfer. Known sets of information for this article include CTU-13, KDDCUP993, and CIC-IDS-20174 (University CTU 2011). Because CTU-13 is easily accessible and has been used in a number of relevant investigations, we chose it over other publicly available datasets. Section 4.1 contains about the dataset. The quantitative information gathering is then evaluated to find patterns and frequency of occurrence. The StartTime's raw data characteristics, TotPkts, TotBytes, SrcBytes, Name, srcAddr, Sports, and Dir netflows.
- Extraction of Features: After picking a data collection, determine and extract the characteristics. This activity is indeed an important process. Because the category attributes of NetFlow information must be written as numbers or Boolean, this results at an overly huge array and memory problems. So we evaluate NetFlow circulation using an elapsed time and a schema to reduce the amount of information that must be analysed. We settled on a 2-minute time limit with a 1-minute move. Following preprocessing of the bi-directional NetFlow dataset, the category and numerical way of attributes that define the data collection are extracted within the designated time frames

Cilt 7, Sayı 1 | Yaz 2023
Volume 7, No 1 | Summer 2023, 11-26

- Feature Choice: It must select the strategy according to the programmes that were deleted. Utilize feature selection techniques to make the input training matrix smaller. Principal Component Analysis (PCA), Backward Feature Elimination Wrapping, Random Forestry Classification embedding approaches, and t-distributor with Stochastic Neighbor Embedding are some of the methods used to achieve this objective (t-SNE).
- Algorithm comparison: Six important computations were put to the test. Deep Learning and Deep Learning techniques (CNN, DNN, LSTM)
- Botnet Detection: The next step in our plan is to evaluate the model against the CTU-13 informational collection to verify if it can reliably detect botnet traffic.

3. Related Works

In recent years, studies on Malware Detection Using Machine Learning and Deep Learning have been examined.

According to Liu et al. (2017), the authors present a machine learning-based system for analyzing malicious software, which comprises three modules: data processing, decision making, and new malicious software detection. The data processing module focuses on extracting the characteristics of malicious software using gray-scale images, Opcode n-grams, and import functions. The decision-making module utilizes these extracted features to classify malicious software and identify suspicious samples. Finally, the detection module employs the shared nearest neighbor (SNN) clustering algorithm to discover new families of malicious software. The proposed approach is evaluated using a dataset of over 20,000 malicious software samples collected by Kingsoft, ESET NOD32, and Anubis. The results demonstrate the effectiveness of the system in classifying unknown malicious software, achieving an accuracy of up to 98.9%. Furthermore, the system successfully detects 86.7% of new malicious software, indicating its capability to identify previously unseen threats.

In their research study, Kedziora et al. (2019) focused on the identification of malware in mobile applications using machine learning methods and reverse engineering of Android Java code. They compiled a collection of 1958 apps and discovered the characteristics of malicious software, including 996 malware applications. To determine the algorithms that provide the highest malware detection rate, they initially selected a distinct set of features. Three attribute selection algorithms and five classification algorithms (Random Forest (RF), K Nearest Neighbors (K-NN), Support Vector Machines (SVM), Naive Bayes, and Logistic Regression) were then examined, revealing that SVM yielded the best results.The research aimed to select the most suitable classification algorithms for malware detection on the Android platform, specifically focusing on the identification of the application features with the highest utility in classifying malware. Among the classification algorithms tested, RF and K-NN performed the best. They achieved the highest scores in terms of the percentage of correctly classified instances, ranging from 80.3% to 80.7% for Java code.

Vinayakumar et al. (2019) discuss the potential of more advanced machine learning algorithms, such as deep learning, to eliminate the need for manual feature engineering in the field of malware detection. However, they highlight that recent research papers have shown limitations in the applicability of these algorithms when trained on

biased data, which hampers their performance in real-world scenarios. The article consists of three main parts. Firstly, a comparison is made between deep learning architectures and traditional machine learning algorithms (MLAs) for malware detection, classification, and categorization using various public and private datasets. Secondly, to eliminate dataset bias, the authors employ different splits of the datasets for training and testing the models in a disjoint manner, considering various timelines. Thirdly, the article introduces a novel image processing method with optimized parameters for both MLAs and deep learning architectures. This method aims to create an efficient model for zero-day malware detection by combining static, dynamic, and image processing techniques within a large-scale data environment.The authors conduct a comprehensive comparative analysis of their proposed deep learning architectures against traditional MLAs. Their results demonstrate the superior performance of the suggested deep learning approaches. This innovative fusion of visualization and deep learning architectures in a hybrid method, incorporating static, dynamic, and image processing, lays the foundation for reliable intelligent zero-day malware detection. The article presents a roadmap for the development of a real-time, scalable, and hybrid deep learning system for visual malware detection.

He and Kim, 2019, The amount of malware assaults is steadily rising every day, driven by financial gains. Malware The initial line of protection against is Detection Systems (MDS).Malware identification is crucial in preventing harmful assaults.technologies that can efficiently and correctly identify malware. Traditional Typically, MDS makes use of conventional machine learning methods.that demand labor-intensive, error-prone feature selection and extraction. Conventionally based deep learningoften employ Recurrent Neural Network (RNN) this is susceptible to repeated API injection. This study demonstrates that a Convolutional Neural Network (CNN)-based Malware Detection System (MDS) has the potential to classify malware files quickly and accurately while being resistant to redundant API injections. By employing a straightforward byte-to-pixel algorithm, malware files can be effectively represented as images, and the results indicate that grayscale images are more resilient to redundant API injection.To enable the network to handle images of any size as input, Spatial Pyramid Pooling has been utilized. However, when dealing with large files, careful handling is required due to physical memory constraints, as the "divide and conquer" method proves to be highly ineffective. On the other hand, simple bi-linear interpolation shows promising results as a resizing method.

Kim at el, 2018, The prevalence of cellphones has significantly increased the amount of malware. Because of their widespread use, Android smartphones are among the smart gadgets that malware targets the most. The unique framework for Android malware detection is proposed in this study. For efficient feature representation on malware detection, their framework employs a variety of features to reflect the characteristics of Android apps from multiple angles. The features are enhanced using our existence-based or similarity-based feature extraction approach.Besides it is suggested to employ a multimodal deep learning approach as a malware detection model. This study using multimodal deep learning for Android malware detection is the first of its kind. We were able to optimise the advantages of including several feature types with our detection model. We conducted a number of trials with a total of 41,260 samples to assess the performance. They evaluated the precision of our model against that of additional deep learning network models. Additionally, we analyzed our approach in a number of ways, including the

Cilt 7, Sayı 1 | Yaz 2023
Volume 7, No 1 | Summer 2023, 11-26

effectiveness of model updates, the value of certain features, and our approach to feature representation. Additionally, we contrasted the comparing the effectiveness of our approach to those existing approaches based on

Alzaylaee et al. (2020) investigate the growing prevalence of Android malware, which has been fueled by the widespread adoption of Android phones. The authors emphasize that conventional techniques for detecting malware have become less effective due to the increasingly sophisticated obfuscation and detection evasion techniques employed by Android malware. In their essay, they propose DL-Droid, a deep learning approach that leverages stateful input creation and dynamic analysis to identify potentially harmful Android applications. The researchers conducted experiments on real devices, using a dataset comprising over 30,000 applications, including both benign and malicious samples. They compared the performance and code coverage of the stateful input generation technique with the widely used stateless approach, which employs a deep learning system. The results demonstrate that DL-Droid surpasses conventional machine learning approaches, achieving detection rates of up to 97.8% when using only dynamic features and 99.6% when combining dynamic and static features. Furthermore, DL-Droid with state-based input creation outperforms existing state-of-the-art techniques.However, the study underscores the importance of enhancing input generation for dynamic analysis, highlighting it as a crucial area for further improvement. These findings contribute to the advancement of effective Android malware detection and reinforce the value of deep learning methodologies in addressing the challenges posed by sophisticated Android malware.

Qiu et al. (2020) explore the use of deep learning (DL) in Android malware detection, considering it a logical choice. The study addresses various challenges faced by academics and practitioners, including DL architecture selection, feature extraction and processing, performance measurement, and the acquisition of high-quality data. They classify the literature based on DL architectures such as Fully Convolutional Network (FCN), CNN, RNN, Deep Belief Network (DBN), Autoencoder (AE), and hybrid models. The focus of the study is to describe code semantics for Android malware detection and highlight the research horizon in this area. The authors also discuss the difficulties encountered in this relatively young discipline and provide their insights on potential future research prospects.

Singh and Singh, 2021 were retrieved utilizing analytic approaches determines how successful the malware detection system is. The research environments may be built up in a variety of ways utilizing both static and dynamic tools. The malware classifiers must be trained in the second step. Traditional approaches were employed in the past, but today machine learning algorithms are used to classify malware since they can keep up with the complexity and speed of virus creation. This paper presents a thorough analysis of machine learning-based malware detection methods. This study also explores several difficulties in creating virus classifiers. The development of an efficient malware detection system is described as a last directive after dealing with numerous malware detection challenges.

Usman at el, 2021 According to study, In the near future, things will need to link to one another, which might lead to the collection of private information and result in numerous security risks and cybercrimes. Innovative cyber security methods that can detect harmful Internet Protocol (IP) addresses before communication are needed to stop cybercrimes. The IP reputation system, which is used to profile the behavior of security risks to the cyber-physical system, is one of the finest methods. Big data forensics is used to anticipate IP reputation at the pre-acceptance stage and classify related zero-day attacks using behavioral analysis and the Decision Tree (DT) approach. The suggested

method draws attention to the large data forensic concerns and simultaneously computes risk score, severity, and confidence and lifespan ratings. The proposed system is assessed in two stages: first, the ML approaches are compared to get the best F-measure, accuracy, and recall scores; second, the overall reputation system is compared to the current reputation systems. In addition to being cross-checked with other sources, our framework is able to lessen security concerns that were disregarded by out-of-date reputation engines that are now in use.

Tobiyama, Yamaguchi, and Shimad discuss the experimental findings obtained after feature reduction using classification models and evaluate them using various metrics including accuracy, recall, specificity, and precision. Table 1 illustrates that VT-based feature reduction, when combined with RF, achieves the highest accuracy of 99.78% among different feature reduction methods. This particular feature set slightly outperforms the cases without reduction and RF, which have an accuracy of 99.74%. When combined with RF, AE-1L outperforms the deep Autoencoder AE-3L and attains the highest accuracy of 99.41%. On the other hand, AE-3L-based reduction performs the worst among all the methods evaluated. VT (and RF) yields the highest True Positive Rate (TPR) of 99.59%, followed by None, while the highest True Negative Rate (TNR) of 100% is achieved without any feature reduction (None and RF). The results indicate that RF performs the best among the various classification models, reaching an accuracy of 99.7% (RF and VT). RF also achieves the second-highest accuracy without any feature reduction. Among different deep learning models, both DNN-3 Land and DNN-7L exhibit an accuracy rate of 98.99% when combined with AE-1L. RF combined with VT and None feature reduction produces the highest TPR and TNR, respectively. In this article, the authors propose a two-stage DNN method for detecting malware processes in infection detection. Their approach involves classifying features extracted from a behavior-based language model using CNN from images. The feature image is created from behavioral features extracted by RNN. The researchers validate their classifier using 150 process behavior logs and employ 5-fold cross-validation. Multiple experiments are conducted under different conditions, and the validation results are compared to identify the best result, which achieves an AUC of 0.96 when the feature image size is 30x30. Component analysis is performed to demonstrate the effectiveness of the features extracted from the RNN trained using the proposed approach. However, due to the limited amount of data available, the authors were unable to leverage large-scale DNNs. Increasing the amount of data is suggested as future work to enhance the approach. Furthermore, augmenting the data quantity to validate the dataset is also considered for future research.

It has been observed in the literature that there are several studies using artificial learning methods. It is generally observed that deep learning methods give better results. For this reason, in this study, malware detection was made with 3 deep learning methods and the method performances were compared.

4. Materials and Methods

4.1. Dataset

The CTU-13 dataset, introduced by CTU University in 2011, is an online dataset commonly utilized for developing algorithms related to botnet detection (Dataset, 2011). Its purpose was to create a collection of network traffic that includes both legitimate and botnet traffic, along with background traffic. In this study, the dataset is used for feature extraction and model training. The CTU-13 dataset consists of 13 captures of different botnet samples, each captured

Cilt 7, Sayı 1 | Yaz 2023
Volume 7, No 1 | Summer 2023, 11-26

in bidirectional NetFlow5 files. The characteristics of the dataset are depicted in Figure 1. The dataset contains a total of 138,048 data points.

Table 2 – Characteristics of the botnet scenarios. (CF: ClickFraud, PS: Port Scan, FF: FastFlux, US: Compiled and controlled by us.)

Id	IRC	SPAM	CF	PS	DDoS	FF	P2P	US	HTTP	Note
1	√	√	√							
2	√	√	√							
3	√			√				√		
4	√				√			√		UDP and ICMP DDoS.
5		√		√					√	Scan web proxies.
6				√						Proprietary C&C. RDP.
7									√	Chinese hosts.
8				√						Proprietary C&C. Net-BIOS, STUN.
9	√	√	√	√						
10	√				√			√		UDP DDoS.
11	√				√			√		ICMP DDoS.
12							√			Synchronization.
13		√		√					√	Captcha. Web mail.

Figure 1. Characteristic of Dataset (Dataset, 2011)

4.2. Deep Learning (DL)

Deep learning is a type of machine learning that trains a computer to perform tasks such as voice recognition, image recognition, or making predictions, just like humans do. Deep learning works on basic parameters related to data, rather than working with equations whose properties are prepared beforehand. It processes this data from many processing layers, identifies patterns and allows the computer to learn on its own (Talan and Aktürk, 2021). Deep learning is a machine learning method used to solve complex problems using artificial neural networks and large data sets. Deep learning is inspired by the neural networks of the human brain and is carried out using multi-layer artificial neural networks. Unlike traditional machine learning methods, deep learning algorithm focuses on automatic discovery of features based on data. Therefore, it does not need human intervention to manually define or select attributes. Deep learning aims to learn patterns in large-scale data sets by taking samples.

4.2.1. Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) are helpful for addressing challenging and significant issues in an increasing variety of areas. Uses for CNN inference have indeed been implemented in safety-critical systems that are vulnerable to soft errors caused by high-energy particles, high temperatures, or abnormal voltage. It is crucial to make sure that the CNN inference process is resistant to soft errors. Because all fractal designs cannot be protected by existing algorithm-based fault tolerance (ABFT) strategies, instruction duplication techniques have a large cost, and error-correcting code is unable to protect computational components, considering CNN consequences, ordinary fault - tolerant techniques are not suitable. The best strategies to protect the CNN inference process against soft errors are the main topic of this study. CNN is now viewed as the model for distributed logic & storage, analogue-programmed multifunctional computing matrices.

Cilt 7, Sayı 1 | Yaz 2023
Volume 7, No 1 | Summer 2023, 11-26

Here are some important formulas used in the design and training of CNNs (Wang et al, 2020, Żurek, Pietroń & Wiatr, 2021).

1. Convolutional layer output shape: The output shape of a convolutional layer can be calculated using the following formulas: Output height = (Input height - Filter height + 2Padding) / Stride + 1 Output width = (Input width - Filter width + 2Padding) / Stride + 1 Output channels = Number of filters

2. Pooling layer output shape: The output shape of a pooling layer can be determined as follows: Output height = (Input height - Pooling size) / Stride + 1 Output width = (Input width - Pooling size) / Stride + 1 Output channels = Input channels

3. Forward propagation in a convolutional layer: The forward propagation in a convolutional layer can be described as follows: z[i, j, k] = (input[m, n, l] * filter[p, q, l, k]) + b[k] a[i, j, k] = activation(z[i, j, k]) Here, i, j, k represent the indices for height, width, and channel of the output feature map, m, n, l represent the indices for height, width, and channel of the input feature map, p, q denote the indices for height and width of the filter, b represents the bias term, and activation is a nonlinear activation function such as ReLU.

4. Backward propagation in a convolutional layer: The backward propagation in a convolutional layer can be defined as follows: dL/dinput[m, n, l] += sum_over_i, sum_over_j, sum_over_k(dL/da[i, j, k] * filter[p, q, l, k]) dL/dfilter[p, q, l, k] += sum_over_i, sum_over_j(dL/da[i, j, k] * input[m + p, n + q, l]) dL/db[k] += sum_over_i, sum_over_j(dL/da[i, j, k]) Here, dL/dinput[m, n, l] represents the gradient of the loss with respect to the input, dL/dfilter[p, q, l, k] denotes the gradient of the loss with respect to the filter, dL/db[k] represents the gradient of the loss with respect to the bias, and dL/da[i, j, k] represents the gradient of the loss with respect to the output feature map.

5. Forward propagation in a pooling layer: The forward propagation in a pooling layer can be expressed as follows: a[i, j, k] = max_over_p, max_over_q(input[Stridei + p, Stridej + q, k]) Here, i, j, k denote the indices for height, width, and channel of the output feature map, p, q represent the indices for height and width of the pooling window, and Stride is the stride of the pooling operation.

6. Backward propagation in a pooling layer: The backward propagation in a pooling layer can be represented as: dL/dinput[Stridei + p, Stridej + q, k] += KroneckerDelta(input[Stridei + p, Stridej + q, k], max_over_p, max_over_q(input[Stridei + p, Stridej + q, k])) * dL/da[i, j, k] Here, dL/dinput[Stridei + p, Stridej + q, k] represents the gradient of the loss with respect to the input of the pooling layer. The KroneckerDelta function is used to determine if the input value at position (Stridei + p, Stridej + q, k) is equal to the maximum value obtained during the forward propagation. If they are equal, the gradient dL/da[i, j, k] is accumulated into the corresponding input position; otherwise, the gradient is multiplied by zero, effectively ignoring the non-maximum values.

Cilt 7, Sayı 1 | Yaz 2023
Volume 7, No 1 | Summer 2023, 11-26

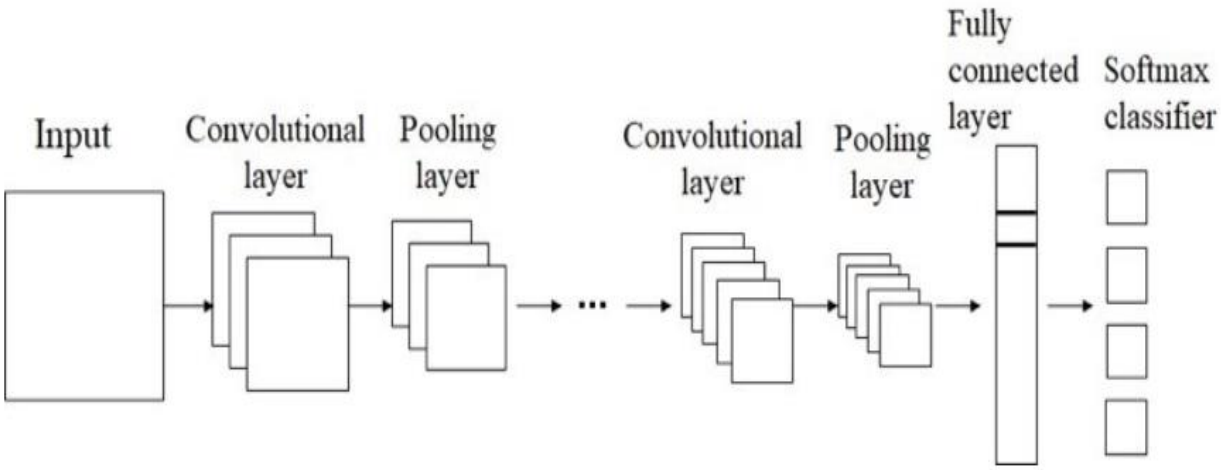

Figure 2. CNN Structure (You et al, 2019)

4.2.2. Multi-Task Deep Neural Networks (DNN)

DNN is a multi-task learning based neural network architecture proposed by Liu et al. in 2019. The DNN architecture consists of three main components: a shared transformer layer, task-specific transformer layers, and a task-specific output layer. The shared transformer layer is a multi-layer bidirectional transformer encoder that learns a shared representation for all the tasks. The task-specific transformer layers are task-specific transformers that are added on top of the shared layer to capture task-specific information. The task-specific output layer consists of task-specific classifiers that produce the final outputs for each task. During training, the DNN model is trained on all the tasks simultaneously using a joint training objective that combines the individual task losses. The joint objective helps the model to learn shared representations that can benefit all the tasks. DNN architecture involves several mathematical formulas for training and prediction (Cheng et al, 2019). Here are some of the key formulas used in DNN:

1. Self-Attention Mechanism: The self-attention mechanism employed in the shared transformer layer of DNN can be defined as follows: Attention(Q,K,V) = softmax(QK^T/sqrt(d_k))V Here, Q, K, and V represent the query, key, and value matrices, respectively, while d_k denotes the dimensionality of the key vectors. The self-attention mechanism calculates a weighted sum of the value vectors based on the similarity between the query and key vectors.

2. Multi-Head Attention: DNN utilizes multi-head attention to enable the model to attend to different parts of the input simultaneously. The multi-head attention mechanism can be defined as follows: MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O head_i = Attention(QW^Q_i,KW^K_i,VW^V_i) In this formulation, Q, K, and V are input matrices, W^Q_i, W^K_i, and W^V_i are learnable matrices specific to the i-th head, and W^O is a learnable matrix for the output projection. The multi-head attention mechanism computes h separate attention functions in parallel and concatenates their outputs.

3. Task-Specific Output Layer: The task-specific output layer in DNN is defined as follows: p_i = softmax(W^{o,i}h_i + b^{o,i}) Here, h_i represents the output of the task-specific transformer layer for the i-th task, W^{o,i} and b^{o,i} are learnable parameters specific to the i-th task, and p_i is the probability distribution over the output classes for the i-th task.

4. Joint Training Objective: DNN adopts a joint training objective that combines the individual task losses as follows: L = sum_i lambda_i L_i In this equation, L_i represents the cross-entropy loss for the i-th task, and lambda_i is a task-specific weight that determines the relative importance of each task in the joint objective. The task weights lambda_i are learned during the training process

. The structure of the DNN algorithm is shown in figure 3.

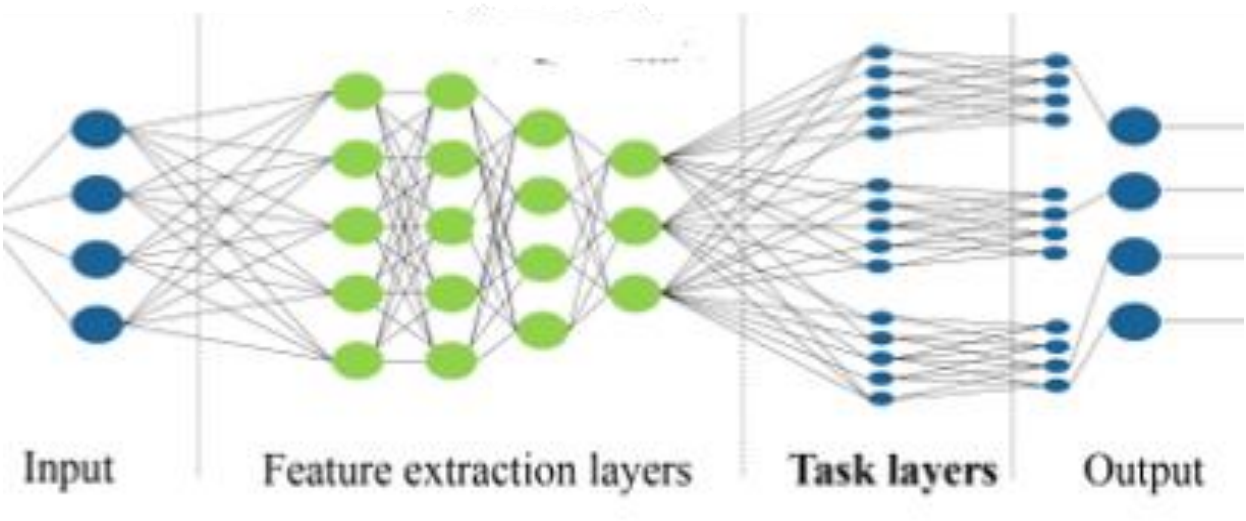

Figure 3. DNN (Ye et al. 2018)

4.2.3. Long Short Term Memory (LSTM)

Long short-term memory networks (LSTM), are a specific type of recurrent neural network (RNN) that excel at recognizing long-term dependencies. They were initially introduced by Hochreiter and Schmidhuber in 1997 and have since been further developed and popularized by various researchers. LSTMs have gained significant popularity and have demonstrated remarkable performance in solving a wide range of problems. The primary purpose of LSTMs is to address the challenge of capturing long-term dependencies. Unlike traditional RNNs that often struggle with this issue, LSTMs are specifically designed to retain information over extended periods. They accomplish this by utilizing recurrent neural network modules that are repeated throughout the network architecture. In a standard LSTM, a single tanh layer typically constitutes the recurrent module. (Olah, 2015).

LSTM are particularly effective in overcoming the vanishing gradient problem that arises in traditional RNNs. They achieve this through the use of various mathematical formulas that govern their behavior and allow them to retain

Cilt 7, Sayı 1 | Yaz 2023
Volume 7, No 1 | Summer 2023, 11-26

and propagate information over longer time intervals. LSTM have several mathematical formulas that govern their behavior, including; (Pathak, Pakray and Das, 2019).

1. Input gate: The input gate within an LSTM cell determines the extent to which new input should be incorporated into the memory cell. It can be defined as follows: i_t = σ(W_i [h_{t-1}, x_t] + b_i) Here, x_t represents the input at time t, h_{t-1} is the previous hidden state, and σ denotes the sigmoid function.

2. Forget gate: The forget gate in an LSTM cell regulates the extent to which the previous memory cell state should be retained. It is defined as: f_t = σ(W_f [h_{t-1}, x_t] + b_f) In this equation, W_f and b_f refer to the weight matrix and bias vector specific to the forget gate, respectively.

3. Cell state: The cell state in an LSTM cell is updated based on the input and forget gates. The update can be formulated as follows: c_t = f_t * c_{t-1} + i_t * tanh(W_c [h_{t-1}, x_t] + b_c) Here, tanh represents the hyperbolic tangent function, while W_c and b_c denote the weight matrix and bias vector associated with the cell state update.

4. Output gate: The output gate in an LSTM cell determines the output of the cell based on the updated cell state. It can be defined as: o_t = σ(W_o [h_{t-1}, x_t] + b_o) h_t = o_t * tanh(c_t) In these equations, W_o and b_o represent the weight matrix and bias vector specific to the output gate, and h_t corresponds to the current hidden state

The structure of the LTSM algorithm is shown in figure 4.

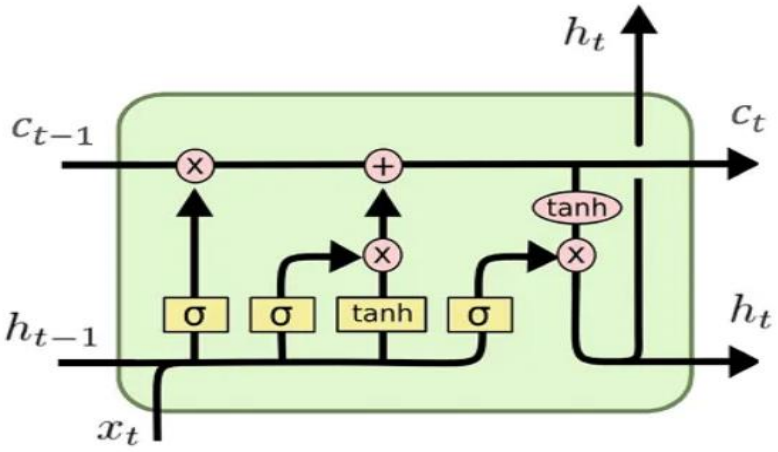

Figure 4. LSTM Standart Structure (Olah, 2015)

4. Results and Discussion

There are several evaluation metrics to check used machine learning algorithms, including accuracy, precision and recall. lThe formulas for these scales are as follows: TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

Cilt 7, Sayı 1 | Yaz 2023
Volume 7, No 1 | Summer 2023, 11-26

- TP (True positive): It means true to truth.
- FP (False positive): False means true.
- TN (True negative): Saying wrong to wrong.
- FN (False negative): Saying true to false

Accuracy: The ratio of correctly predicted samples to all samples, or simply the ratio of correctly predicted samples to all samples, is the most intuitive performance metric. Accuracy is shown in formula 1.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

Precision: is the ratio of correctly predicted positive samples to the total predicted positive samples. Precision is shown in formula 2.

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

Recall: is the proportion of accurately anticipated positive samples to the total number of positive samples predicted. Recall is shown in formula 3.

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

Measurement of performance the percentage of cases that were correctly identified are accuracy, precision abd recall. This study was carried out in visual studio 2022 python 3.10.6. program on a 256 SSD capacity computer with 8GB RAM. The results obtained in this study are shown in Table 1 and Figure 5.

Tablo 1. performance of deep learning algorithms

Algorithms	Accuracy	Precision	Recall
LSTM	0,982	0.988	0.990
CNN	0,965	0.969	0.978
DNN	0,933	0.939	0.940

The findings obtained as a result of this study are explained in articles below.

- Since the result of each method is over 90%, all methods are considered successful.
- The best results in Malware detection were obtained from the LTSM, CNN and DNN methods, respectively.

- With 3 deep learnıng algorıthms, average 96% in Accuracy, average 97% in Precision, average %97 in Recall was obtained.
- LSTM algorithm Accuracy gave better results than 4.9% DNN algorithm and 1.7% CNN algorithm.
- LSTM algorithm Precision gave better results than 4.9% DNN algorithm and 1.9% CNN algorithm.
- LSTM algorithm Recall gave better results than 5% DNN algorithm and 1.2% CNN algorithm.

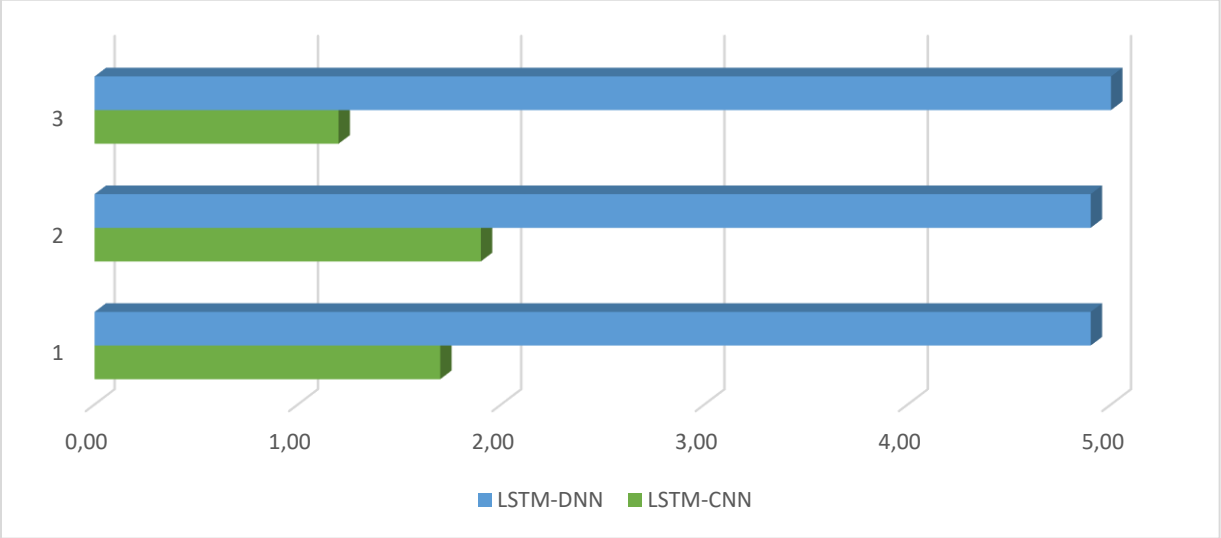

Figure 5. Comparison of LSTM algorithm with CNN and DNN

5. Conculusion

In this study aim to detect malware using the dataset created by CTU University 2011 to generate certified botnet traffic blended with normal traffic and infrastructure traffic. Classification was made with three different deep learning algorithms (LSTM, CNN, DNN) and the results were compared. The LSTM algorithm has been found to be the most effective method for malware detection. Three different measurement parameters (accuracy, precision, recall) were used, and the highest result was obtained with the LSTM algorithm in each of them. As a result of this study, it has been observed that deep learning algorithms give good results in malware detection. In this study, the most effective results were found as Accuracy 0.982, Precision 0.988 and recall 0.990.

Referances

Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89, 101663.

Aycock, J. (2006). Computer viruses and malware (Vol. 22). Springer Science & Business Media.

Cheng, Y., Fu, S., Tang, M., & Liu, D. (2019). Multi-task deep neural network (MT-DNN) enabled optical performance monitoring from directly detected PDM-QAM signals. Optics express, 27(13), 19062-19074.

Danniels, M. "Denial of Service Attacks," Imperva, 2015. [Online]. Available: https://www.incapsula.com/ddos/ddos-attacks/denial-of-service.html. [Accessed 17 Juny 2015]

He, K., & Kim, D. S. (2019, August). Malware detection with malware images using deep learning techniques. In 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) (pp. 95-102). IEEE.

Kedziora, M., Gawin, P., Szczepanik, M., & Jozwiak, I. (2019). Malware detection using machine learning algorithms and reverse engineering of android java code. International Journal of Network Security & Its Applications (IJNSA) Vol, 11.

Kim, T., Kang, B., Rho, M., Sezer, S., & Im, E. G. (2018). A multimodal deep learning method for android malware detection using various features. IEEE Transactions on Information Forensics and Security, 14(3), 773-788.

Liu, L., Wang, B. S., Yu, B., & Zhong, Q. X. (2017). Automatic malware classification and new malware detection using machine learning. Frontiers of Information Technology & Electronic Engineering, 18(9), 1336-1347.

Mao, W., Cai, Z., Towsley, D., Feng, Q., & Guan, X. (2017). Security importance assessment for system objects and malware detection. *Computers & Security*, *68*, 47-68.

Olah, C., (2015). Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, Erişim tarihi 7 Ekim 2020

Pathak, A., Pakray, P., & Das, R. (2019, February). LSTM neural network based math information retrieval. In 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP) (pp. 1-6). IEEE.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., & Xiang, Y. (2020). A survey of android malware detection with deep neural models. ACM Computing Surveys (CSUR), 53(6), 1-36.

Singh, J., & Singh, J. (2021). A survey on machine learning-based malware detection in executable files. Journal of Systems Architecture, 112, 101861.

Talan, T. ve Aktürk, C. (2021) Bilgisayar Biliminde teorik ve uygulamalı araştırmalar, Efe akademi yayıncılık, Istanbul, ISBN: 978-625-8065-42-8

Tobiyama, S., Yamaguchi, Y., Shimada, H., Ikuse, T., & Yagi, T. (2016, June). Malware detection with deep neural network using process behavior. In *2016 IEEE 40th annual computer software and applications conference (COMPSAC)* (Vol. 2, pp. 577-582). IEEE.

Usman, N., Usman, S., Khan, F., Jan, M. A., Sajid, A., Alazab, M., & Watters, P. (2021). Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics. Future Generation Computer Systems, 118, 124-141.

Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., & Zheng, Q. (2020). IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. Computer Networks, 171, 107138.

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., & Venkatraman, S. (2019). Robust intelligent malware detection using deep learning. IEEE Access, 7, 46717-46738.

Wang, Z. J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., ... & Chau, D. H. P. (2020). CNN explainer: learning convolutional neural networks with interactive visualization. IEEE Transactions on Visualization and Computer Graphics, 27(2), 1396-1406.

Ye, Z., Yang, Y., Li, X., Cao, D., & Ouyang, D. (2018). An integrated transfer learning and multitask learning approach for pharmacokinetic parameter prediction. Molecular pharmaceutics, 16(2), 533-541

You, W., Shen, C., Wang, D., Chen, L., Jiang, X., & Zhu, Z. (2019). An intelligent deep feature learning method with improved activation functions for machine fault diagnosis. IEEE Access, 8, 1975-1985.

Soofi, A. A., & Awan, A. (2017). Classification techniques in machine learning: applications and issues. J. Basic Appl. Sci, 13, 459-465.