# Android Malware Application Detection using Multi-layer Perceptron
# Çok Katmanlı Algılayıcı Kullanarak Android Kötü Amaçlı Yazılım Uygulama Tespiti

Gokhan Altan[1]*, Furkan Pasalioglu[1]
[1] Department of Computer Engineering, Iskenderun Technical University, Hatay, Turkey
ORCIDs: 0000-0001-7883-3131, 0000-0002-7630-5375
E-mails: gokhan.altan@iste.edu.tr, furkanpasalioglu@yahoo.com
*Corresponding author.

***Abstract*—Cyber-attacks are one of the most critical problems that seriously threaten society. Whereas there are various presentations and ways of carrying out cyber-attacks, numerous mechanisms and techniques exist to defend applications. Many malware creators have chosen the Android operating system as a target due to its popularity. Thousands of new malware samples, aiming to infect new devices daily, are trying to circumvent the security measures implemented by Android app stores. This study experiments with a multi-layer perceptron model for Android malware detection. This proposed system is based on static analysis techniques on Android. We analyzed popular machine learning algorithms with a total number of 129013 applications (5560 malicious and 123453 harmless software). We achieved higher malware-detection rates of 97.60% in the iterations.**

*Keywords—Classification; multi-layer perceptron; malware detection; cyber-attack; cyber-security; Android*

***Özetçe*—Siber saldırılar, toplumu ciddi şekilde tehdit eden son zamanların en kritik sorunlarından biridir. Siber saldırıları gerçekleştirmenin çeşitli sunumları ve yolları olsa da, uygulamaları savunmak için çok sayıda mekanizma ve yöntem mevcuttur. Birçok kötü amaçlı yazılımcı, popülaritesi nedeniyle Android işletim sistemini hedef olarak seçmektedir. Her gün yeni cihazlara erişmeyi amaçlayan binlerce yeni kötü amaçlı yazılım örneği, Android uygulama mağazalarının uyguladığı güvenlik önlemlerini atlatmaya çalışmaktadır. Bu çalışma, Android tabanlı çalışan kötü amaçlı yazılım tespiti için çok katmanlı algılayıcı modeli ile deneyler gerçekleştirmektedir. Önerilen bu sistem Android üzerinde kullanılan statik analiz tekniklerine dayanmaktadır. Popüler makine öğrenimi algoritmalarını toplam 129013 uygulama (5560 kötü amaçlı ve 123453 zararsız yazılım) ile analiz edilmiştir. İterasyonlarda %97,60'dan daha yüksek kötü amaçlı yazılım algılama oranları elde edilmiştir.**

*Anahtar Kelimeler—Sınıflama; çok katmanlı algılayıcılar; kötü amaçlı yazılım tespiti; siber atak; siber güvenlik; Android*

## I. Introduction

Cyber-attacks are the most common ways for hackers to take over the whole or local controls of user and device specifications. The widespread cyber-attacks focus on the ways that are widespread regarding user and software vulnerability. However, there are numerous mechanisms and techniques to deal with them. Malware has undergone various changes since its first release and has become a more complex structure for developers and cyber-attackers.

Android operating system is frequently used on different platforms, especially mobile devices. The overwhelming preference for the Android operating system has made it an attractive target for attackers [1]. The open-source Android platform may cause it to be considered insecure in applications. Developers can easily access the Android source code and publish various applications for the Android operating system. This circumstance may cause some security vulnerabilities. In addition, delays in security updates released by device manufacturers also allow attacks [1].

There are many types of malware that target the Android operating system. This malicious software can be listed as spyware, backdoor, trojans, worms, ransomware, and botnets [2]–[4]. Spyware is software that collects user-sensitive data, tracks users, and shares this data with malicious people. MobileSpy is the first professional spyware released on the Android operating system [3]. Nowadays, spyware, which has many alternatives, is preferred by politicians, people in business, governments, and more. Spyware can be used to obtain GPS locations, correspondence, and bank information of individuals, as well as to obtain consumer habits for customized advertisements [2]. Moreover, advanced artificial intelligence algorithms are adapted to sensing systems in IIoT networks for detecting malicious applications and various cyber-attack from network traffic [17].

Backdoors are basically software that provides unauthorized access to attackers, allowing unauthorized operations on infected devices. Hummingbad, Damon, and FakeLook are some backdoor scripts published for the Android operating system [5].

Trojans are designed to gain the trust of users. Therefore, they are usually prepared by analogy with reliable application designs, logos, and colors. Social engineering is also used when designing trojan horses. Mobile applications and other methods can infect trojans, such as movie or music files or fake device updates. Trojan damages privacy and cause data to be stolen.

Ransomware is software that encrypts files stored on devices until payment is made. The encryption key is needed to open files for use again. ScarePackage ransomware spread to thousands of devices within 30 days and victimized many people [4].

Botnets are large-scale network attacks that allow attackers to control devices [6]. Botnets can be controlled by a Botmaster and manipulate the device for various purposes. Botnets can be used for Distributed Denial of Service (DDoS) attacks, sending premium messages or spamming, and mining applications for digital currencies. WireX malware spread to more than 120,000 Android devices in 2017 and was used for DDoS attacks [7], [8].

Various studies offer malware detection and classification mechanisms by utilizing machine learning techniques. These studies address the problem from two perspectives, depending on static and dynamic analysis approaches. Static analysis has been the subject of a significant amount of research due to the fact that information is easy to obtain and contains information that indicates application operations and intentions. In these studies, package names, API calls, permissions, and metadata are used as inputs to train classification models.

In a study conducted in 2017, an automated system based on sequence classification was developed using deep learning techniques. Their system created API management calls. The malware infections were detected correctly, with an f1-score between 96% and 99%. In addition, the malware was associated with virus families, with a false positive rate of 0.05%-2% [9]. In another study in 2017, a permission-based Convolutional Neural Network (CNN) model was utilized to detect malware. They used a total number of 2500 applications, 2000 malware, and 500 harmless software. Thanks to this dataset, the malware was caught with an accuracy rate of 93% [10].

In 2018, a multi-modal deep learning method was proposed in the study using 41260 samples. This proposed study used an entity-based or similarity-based feature extraction method [11]. In another study in 2018, a deep learning-based system was proposed using a control flow graph, data flow graph, and their possible combinations. These flow graphs are coded to form a matrix, and the classification model is trained via CNN [12].

In 2019, a new method based on deep learning mechanisms was proposed for malware detection. This proposed study examines system calls as a kind of natural language, and a sensor model is constructed using the Long Short-Term Memory (LSTM) model [13]. In another study, the API call graph presents all possible execution paths malware could follow during runtime. They utilized a total number of 33139 malware and 25000 benign software in the proposal and obtained an accuracy rate of 98.86% with different graphing algorithms and network configuration parameters [14].

In 2020, a study based on deep learning was published using more than 30000 applications. In their proposal, accuracy rates of 97.8% and 99.6% were achieved using only dynamic analysis and both dynamic-static analysis [15]. Another study in 2020 proposed a two-layer method for detecting malware. In the first layer, static features are combined with a fully connected neural network, and an accuracy of 95.22% is achieved. In the other layer, applications are examined over network traffic. The detection rate was presented with an accuracy rate of 99.3% using cascading CNN and AutoEncoder [16].

On the other hand, image-based features and deep learning techniques were also discussed together. In this context, the features obtained from the application were converted to grayscale images. Four images were obtained for each application, and a detection accuracy rate of over 98% was achieved [18]. In another study, an adjacency matrix was created for each application and sent to the CNN model as input. They reached a malignant detection rate of 98% and a malware family detection rate of 97% [19].

In 2022, a model was presented to analyze the behavior of malicious applications based on API call graphs using CNN. In this study, API call graphs were examined to increase efficiency, and classification was made using similarity between graphs. In this proposed model, a success rate of 91.27% was achieved [20]. Another study presented a behavioral Android malware detection model. The LSTM model is used to classify snapshots of reconstructed API and system call sequences in the model based on various static and dynamic properties. This study achieved a competitive accuracy rate, especially against ransomware [21].

## II. Material and Methods

### A. DREBIN Dataset

The Drebin dataset consists of 123,453 benign and 5,560 malware. The features in the Drebin data set can be obtained by static analysis. Every app developed for Android should contain a manifest file named AndroidManifest.xml that contains data supporting the application installation and subsequent running. In addition, Android applications are developed in programming languages such as Java and Kotlin and compiled into bytecode optimized for the Dalvik virtual machine. Information about API calls and data in an application can efficiently be obtained by extracting the bytecode from .dex files. The data obtained by static analysis from the manifest file and dex codes were collected under eight feature sets. These eight features are described below.

*Hardware components*: This initial feature contains the requested permissions to access hardware components. It may cause certain security implications, as using certain combinations of hardware often reflects malicious behavior.

*Requested permissions*: The permission system is one of the most important Android security mechanisms. The user actively grants permissions during installation or when the application is launched after Android 6.0, allowing an application to access security-related resources.

*Application components*: An android application can reach four components to grant system activities interfaces. They are activities, services, content providers, and recipients. Each component was collected as the name lists. According to the blocklist, each name was manipulated as malicious and normal to identify well-known malware components.

*Filtered intents*: Inter-process and intra-process communication in Android is mainly done with intent. Because malware usually listens for specific intents, all intents listed in the manifest have been collected as another set of features.

*Restricted API calls*: The Android permission system restricts access to several critical API calls. A particular case that introduces malicious behavior is the restricted API calls where the required permissions are not requested.

*Used permissions*: All of the calls issued in restricted API calls can be used as grounds to determine the subset of both requested and used permissions. Unlike the restricted API calls, this feature set provides a more general view of an application's behavior, as multiple API calls can be protected with a single permission.

*Suspicious API calls*: Certain API calls similar to malware have the possibility to reach private data and resources in the mobile device. Because the requested calls can treat as particularly malicious behavior, it is collected as a separate feature in the Drebin dataset.

*Network addresses*: The malicious applications regularly utilize network connections. It may receive and requests commands or leak private data using an internet connection. Therefore, all IP addresses, host names, and URLs requested by the Android application are included as a feature set. Some of these addresses may be included in botnets.

Since most learning methods work on numeric vectors, the extracted features were manipulated in a vector space.

### *B. The Multi-Layer Perceptron (MLP)*

The Multi-Layer Perceptron (MLP) is a feed-forward neural network algorithm suitable for classification and regression. This network can be expressed as follows:

$$y = f\left(\overrightarrow{x}\right) \tag{1}$$

Where $y \in R^m$ is the expected output of the network and $\overrightarrow{x} \in R^n$ is the input. The building block of MLP is perceptron. A perceptron is a mathematical model used to represent a neuron. Based on the input $\overrightarrow{x} \in R^n$ and its parameters and the weights $\overrightarrow{\omega} \in R^n$ and bias $b \in R$, the internal potential of the sensor is calculated as follows:

$$\zeta = \overrightarrow{\omega} \bullet \overrightarrow{x} + b = \sum_{i=1}^{n} \left(\omega_i \bullet x_i\right) + b = \sum_{i=0}^{n} \left(\omega_i \bullet x_i\right) \tag{2}$$

Based on the internal potential, the output of the y detector is expressed as:

$$y = a\left(\zeta\right) \tag{3}$$

using the activation function "a". Here, functions such as Rectified Linear Units (ReLU), Scaled Exponential Linear Units (SeLU), and hyperbolic tangent can be used as activation functions [22].

The sensors can be organized in layers to form a feed-forward network. An example of such a network is shown in Figure 1. Networks usually have three possible types of layers. These are an input layer, hidden layers, and an output layer. Except for the input layer, the sensors of each layer are connected to all sensors in the previous layer. As a result, the output function $f(\bullet)$ of the network can be thought of as an indexed, recurrent function, given below, which these equations hold:

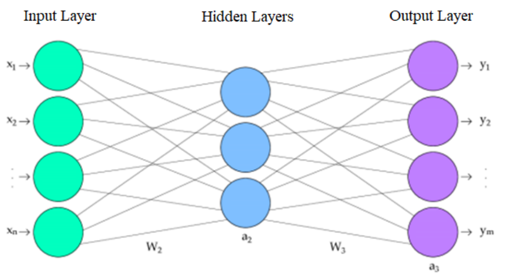

Figure 1: An MLP architecture with three layers and three neurons in the hidden layer

$$y = f_L\left(\overrightarrow{x}\right) \tag{4}$$

$$f_l\left(\overrightarrow{x}\right) = a_l\left(W_l \bullet f_{l-1}\left(\overrightarrow{x}\right)\right) \tag{5}$$

$$f_1\left(\overrightarrow{x}\right) = \overrightarrow{x} \tag{6}$$

Where $a_l$ and $f_l$, represent the activation function and output function for the $l^{th}$ layer, and $W_l$, represents the matrix of weights to the $l^{th}$ layer. We can also show the output and internal potential of each layer in the given order as follows:

$$y_l = f_l\left(\overrightarrow{x}\right) = a_l\left(W_l \bullet y_{l-1}\right) \tag{7}$$

$$\zeta_l = W_l \bullet y_{l-1} \tag{8}$$

The goal of training the network is to minimize the error in the training data by adjusting the weights. Therefore, a cost function C, such as cross-entropy, is used to calculate the error. To find the optimal weights, $\frac{\partial C}{\partial W_l}$ must be calculated.

The main advantage of MLPs is that they scale well with more training data and have expressive power. But they are black boxes, and it isn't easy to find the optimal configuration with many possible architectures and hyperparameter tuning [23], [24], [25].

## III. Experimental Results

Designing malware detection tools requires representative groups of both benign and malware. The classifier uses the feature vector of the application to be analyzed as input. It gives the probability of belonging to a group. In this study, the application to be analyzed first is processed precisely, and feature vectors are created by static analysis. We calculated statistical test metrics, including overall accuracy, precision, recall, f-measure, and true negative rate, for the proposed MLP models to evaluate the system malware detection performance [26], [27].

Static analysis tries to explain the behavior of mobile applications by checking for a set of predetermined features.

| Threshold | Malware Detection Performances in terms of various threshold values (%) | | | | | | |
|---|---|---|---|---|---|---|---|
| | Accuracy | Precision | Recall | f-measure | False Positive Rate (FPR) | False Negative Rate (FNR) | True Negative Rate (TNR) |
| 0.4 | 97.00 | 93.12 | 34.53 | 50.38 | 0.11 | 65.46 | 99.88 |
| 0.5 | 97.26 | 85.76 | 44.85 | 58.90 | 0.34 | 55.14 | 99.65 |
| 0.6 | 97.32 | 72.58 | 64.11 | 68.08 | 1.12 | 38.88 | 98.87 |
| 0.7 | 97.38 | 90.18 | 43.91 | 59.06 | 0.21 | 56.08 | 99.78 |
| 0.8 | 97.51 | 83.67 | 54.09 | 65.70 | 0.48 | 45.91 | 99.51 |
| **0.9** | **97.68** | **77.93** | **64.47** | **70.57** | **0.82** | **35.52** | **99.17** |

Table I: Perception model performance at different threshold values

From this assumption, a depiction of behavior that can be attributed to malicious or harmless software of a particular subset of features will emerge. Properties obtained by static analysis can be expressed as numerical and/or binary variables. After extracting the feature set, such as requested permissions, initiated API calls, or intent, the resulting data is used to create the feature vector.

These feature vectors are given as input to the deep neural network, as indicated in Figure 2. The MLP model with multiple hidden layers is used in the proposal. After composing the hidden layers, the generated hidden vectors are transferred to a ReLU activation layer to detect the relationship between features in the classification. The ReLU layer adds the calculated weights and biases and assigns a probability value for each class. It takes the higher probability result to assign the classification result.

However, the ReLu layer may produce low values in case the feature vectors do not provide enough data to provide unreliable results. In the model created to provide reliable detection results, a threshold is added above the ReLU layer, and inputs below this threshold are classified with a temporary label called ambiguous.

The detection performances for the proposed MLP models with different threshold values are presented in Table I. When the threshold value was set to 0.4, an accuracy rate of 97.00% and a true negative rate of 99.88% were reached. When the threshold value was increased from 0.5 to 0.9, the malware detection accuracy increased by 97.26% to 97.68%. It has filtered out unreliable classification and provides higher performance.

## IV. Conclusion

Mobile devices have become ubiquitous in our lives, providing almost the same functions as personal information processing systems. In recent years, malware developers primarily target the Android platform for different reasons. The contribution of this study to the literature is that it presents a new detection system with the help of the multi-layer perceptron model, using the information obtained by static analysis. The Drebin dataset can detect malicious scripts with an accuracy rate of 97.68% in the proposed model.

## Acknowledgements

The authors thank Deeptech Engineering Company (www.deeptechmuhendislik.com) for providing full access to GPU servers for machine learning analysis.

## Author Contributions

This paper is a part of *Furkan PASALIOGLU*'s MSc thesis. *Dr. Gokhan ALTAN* is the thesis advisor. All authors equally contributed to machine learning and data manipulation applications in the paper.

## References

[1] AV-TEST, The Independent IT-Security Institute. Security Report: Facts and Figures. Magdeburg, 2020.

[2] Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution. *2012 IEEE Symposium on Security and Privacy* 2012; pp. 95-109.

[3] Castillo CA. Android malware past, present, and future. *White Paper of McAfee Mobile Security Working Group 1* 2011; p. 16.

[4] Andronio N, Zanero S, Maggi F. Heldroid: Dissecting and detecting mobile ransomware. *International workshop on recent advances in intrusion detection*, 2015; pp. 382-404.

[5] Martinelli F, Mercaldo F, Nardone V, Santone A, Vaglini G. Model checking and machine learning techniques for HummingBad mobile malware detection and mitigation. *Simulation Modelling Practice and Theory* 2020; 105: 102169.

[6] Parvez F. Android security: A survey of issues, malware penetration, and defenses. *IEEE Communications Surveys and Tutorials* 2015; 17(2): 998-1022.

[7] Shankar S. The do you knows of DDoS attacks. 18 Mart 2021. [Online]. Retrieved from https://www.mcafee.com/blogs/consumer/mobile-ddos/

[8] Douligeris C, Mitrokotsa A. DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks* 2004; 44(5): 643-666.

[9] Karbab EB, Debbabi M, Derhab A, Mouheb D. Android malware detection using deep learning on API method sequences. *Elsevier Digital Investigations Journal* 2017; p. 17.

[10] Ganesh M, Pednekar P, Prabhuswamy P, Nair DS, Park Y, Jeon H. CNN-based Android malware detection. In *International Conference on Software Security and Assurance (ICSSA)*, Altoona, PA, USA, 2017.

[11] Kim T, Kang B, Rho M, Sezer S, Im EG. A multimodal deep learning method for Android malware detection using various features. *IEEE Transactions on Information Forensics and Security* 2019; 14(3): 773-788.

[12] Xu Z, Ren K, Qin S, Craciun F. CDGDroid: Android malware detection based on deep learning using CFG and DFG. Book chapter in *Formal Methods and Software Engineering*, 2018, pp. 5-11.

[13] Xiao X, Zhang S, Mercaldo F, Guangwu H, Sangaiah AK. Android malware detection based on system call sequences and LSTM. *Multimedia Tools and Applications* 2019; 78: 3979–3999.

[14] Pektas A, Acarman T. Learning to detect Android malware via opcode sequences. *Neurocomputing* 2019; 396: 599-608.

[15] Alzaylaee M K, Yerima S Y, Sezer S. DL-Droid: Deep learning based android malware detection using real devices. *Computers and Security* 2020; 89: 101663.

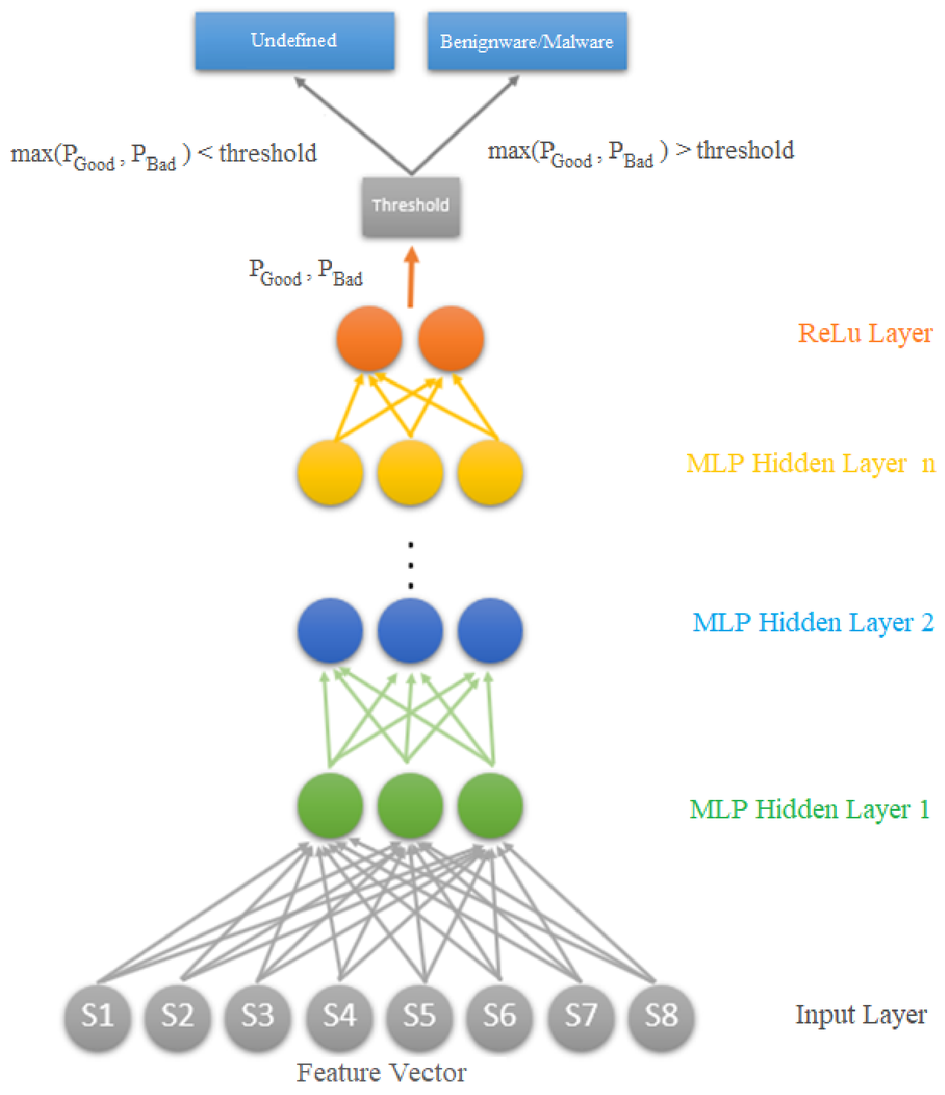

Figure 2: Identification and classification phase of the created system

[16] Feng J, Shen L, Chen Z, Wang Y, Li H. A two-layer deep learning method for Android malware detection using network traffic. *IEEE Access* 2020; 8: 125786-125796.

[17] Altan G. SecureDeepNet-IoT: A deep learning application for invasion detection in industrial Internet of Things sensing systems. *Transactions on Emerging Telecommunications Technologies* 2021; 32(4): e4228.

[18] Bakour K, Unver HM. DeepVisDroid: Android malware detection by hybridizing image-based features with deep learning techniques. *Neural Computing and Applications* 2021; 33: 11499–11516.

[19] Vu LN, Jung S. AdMat: A CNN-on-matrix approach to Android malware detection and classification. *IEEE Access* 2021; 9: 39680-39694.

[20] Kim J, Ban Y, Ko E, Cho H, Yi JH. MAPAS: A practical deep learning-based android malware detection system. *International Journal of Information Security* 2022; 21: 725-738.

[21] Amer E, El-Sappagh S. Robust deep learning early alarm prediction model based on the behavioural smell for android malware. *Computers and Security* 2022; 116: 102670.

[22] Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge: MIT Press, 2016.

[23] Burkov A. The Hundred-Page Machine Learning Book Canada, 2019.

[24] Altan G, Inat G. EEG-based spatial attention shifts detection using time-frequency features on empirical wavelet transform. *Journal of Intelligent Systems with Applications*, 2021, 4 (2):144-149,.

[25] Bulut E, Ozturk G, Kaya I. Classification of sleep stages via machine learning algorithms. *Journal of Intelligent Systems with Applications* 2022; 5(1): 66-70.

[26] Pehlivan S, Isler Y. Detection of heart disease risk utilizing correlation matrix, random forest and permutation feature importance approaches. *Journal of Intelligent Systems with Applications* 2020; 3(1): 29-34.

[27] Sayilgan E, Yuce YK, Isler Y. Frequency recognition from temporal and frequency depth of the brain-computer interface based on steady-state visual evoked potentials. *Journal of Intelligent Systems with Applications* 2022; 4(1): 68-73.